	L #	Hits	Search Text
1	L1	1244	(abbreviat$3 compress$3 compact$3) near5 (instruction opcod
2	L2	193	(expand$3 decompress$3 convert$3) near50 1
3	L3	29	(recover$3 recreat$3 transform$3 translat$3) near50 1
4	L4	68	(pipelin$3 stage cycle) near50 1
5	L5	57	(two several couple) near10 1
6	L6	133	(first second) near10 1
7	L7	2	dual near10 1
8	L8	821	(base near20 (offset displacement)) near20 table
9	L9	933	(base near20 (offset displacement)) near20 memory
10	L10	87776	(traslat$3 conver$4 map$4 expan$4) near10 (table memory)
11	L11	770	(8 9) and 10
12	L12	82	(8 9) near99 10
13	L13	82	(8 9) near50 10
14	L14	141	(traslat$3 conver$4 map$4 expan$4) near50 (8 9)
15	L15	30840	(traslat$3 conver$4 map$4 expan$4) adj3 (table memory)
16	L16	43	(base near20 (offset displacement)) near20 15
17	L17	1797	(base near20 (offset displacement)) near20 (index$3 select$3 e
18	L18	8	17 near20 15
19	L19	28043	(traslat$3 conver$4 map$4 expan$4) near10 code
20	L20	3	19 near50 (8 9)

63 62 61 60 59 58 57 56 55 54 53 52 51 50 49 48 47 46 45 44 43 42 41 40 39 38 37 36 35 34 33 32 31 0

															DATA UNIT FORMAT
1	Class	A	8-bit ALU code	dst	src1	0	immed.	Parallel Transfers							A
1	Class	A	8-bit ALU code	dst	src1	1 0	-	scr2	Parallel Transfers						B
1	Class	A	8-bit ALU code	dst	src1	1 1	dstbank	s1bnk	cond.	32-bit imm.					C
1	- - -	1	-	0	-	0	-	0	-	0	- - - - - - - -	0	operation	Parallel Transfers	D
0 1 1	oper	src3	dst2	dst1	src1	src4	src2	Parallel Transfers							E
0 0	Reserved														
0 1 0	Reserved														

38 37 36 35 34 33 32 31 30 29 28 27 26 25 24 23 22 21 20 19 18 17 16 15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0

																			DATA TRANSFER FORMAT
Lmode	d	e	siz	s	La	Gim/x	L	Obank	L	Gmode	reg	e	siz	s	Ga	Lim/x			1
Lmode	o	a	n	0	0 0	0	La	Gim/x	bank	L	Gmode	reg	e	siz	s	Ga	Lim/x		2
Lmode	d	e	siz	s	La	0	Lrm	dstbank	L	0 0	0 0	src	srcbank	dst	Lim/x				3
Lmode	d	e	siz	s	La	0	itm	dstbank	L	0 0	0 1	src	e	siz	D	dst	Lim/x		4
Lmode	reg	e	siz	s	La	1	Lrm	bank	L	0 0	Local Long Offset/x								5
0 0	Global Long Offset/x	bank	L	Gmode	reg	e	siz	s	Ga	0	Grm								6
Lmode	d	e	siz	s	La	0	Lrm	Adstbank	L	0 0	1	- - - -	As1bank	- - -	Lim/x				7
0 0	-	cond.	c	r	g	N C V Z	0	- -	dstbank	-	0 0	0 0	src	srcbank	dst	- - -			8
0 0	-	cond.	c	r	g	N C V Z	0	itm	dstbank	-	0 0	0 1	src	e	siz	D	dst	- - -	9
0 0	-	cond.	c	r	g	N C V Z	Gim/x	bank	L	Gmode	reg	e	siz	s	Ga	1	Grm		10
0 0	-	cond.	c	r	-	N C V Z	0	- -	Adstbank	-	0 0	1	- - - -	As1bank	- - - - - -	R			11

FIG. 43

	Document	U	Title	Current OR
1	USD 6049667 A	□	Computer system, method of compiling and method of accessing address space with pointer of different width therefrom	717/5
2	US 5706481 A	□	Apparatus and method for integrating texture memory and interpolation logic in a computer system	345/519
3	US 5706461 A	□	Method and apparatus for implementing virtual memory having multiple selected page sizes	711/203
4	US 5555387 A	□	Method and apparatus for implementing virtual memory having multiple selected page sizes	711/209
5	US 5548709 A	□	Apparatus and method for integrating texture memory and interpolation logic in a computer system	345/510
6	US 5469431 A	□	Method of and apparatus for channel mapping with relative service identification	370/254
7	US 5412766 A	□	Data processing method and apparatus for converting color image data to non-linear palette	345/431
8	US 4965771 A	□	Printer controller for connecting a printer to an information processor having a different protocol from that of a printer	358/1.13

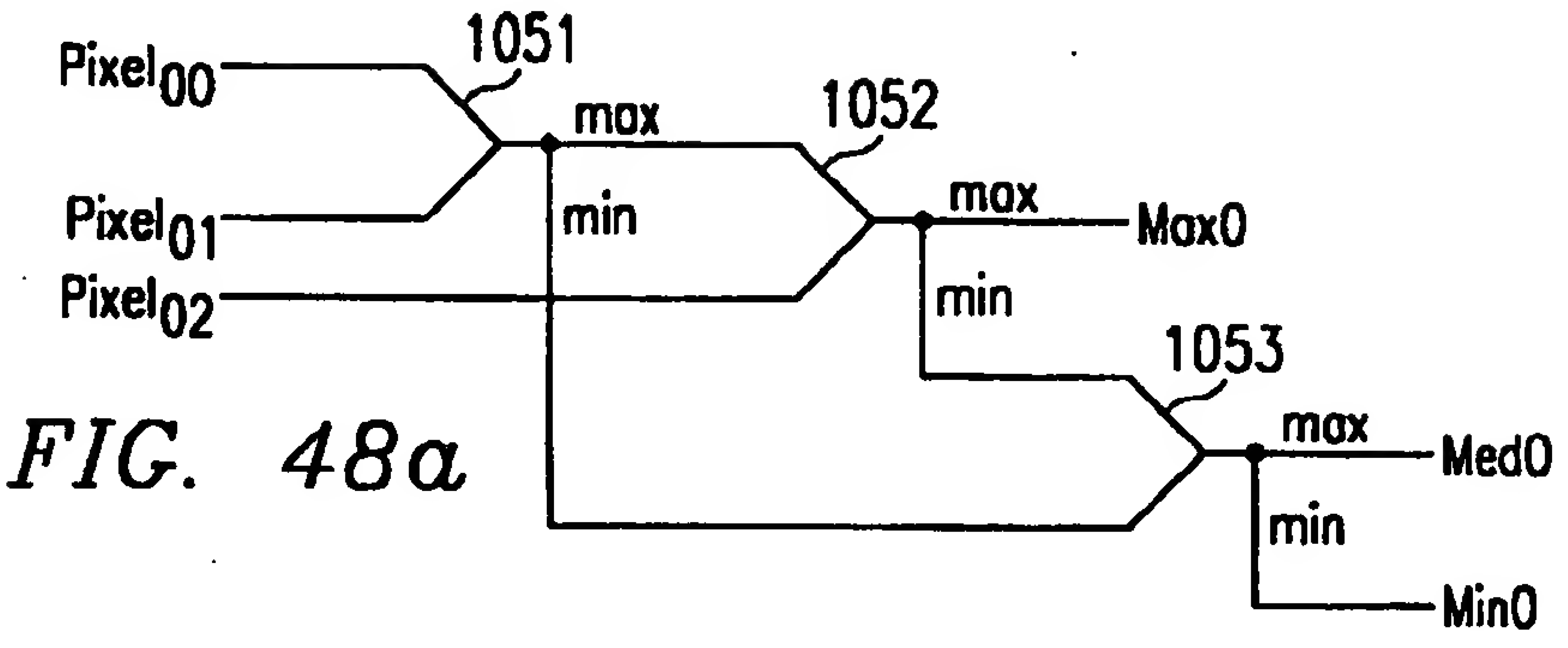


FIG. 48a

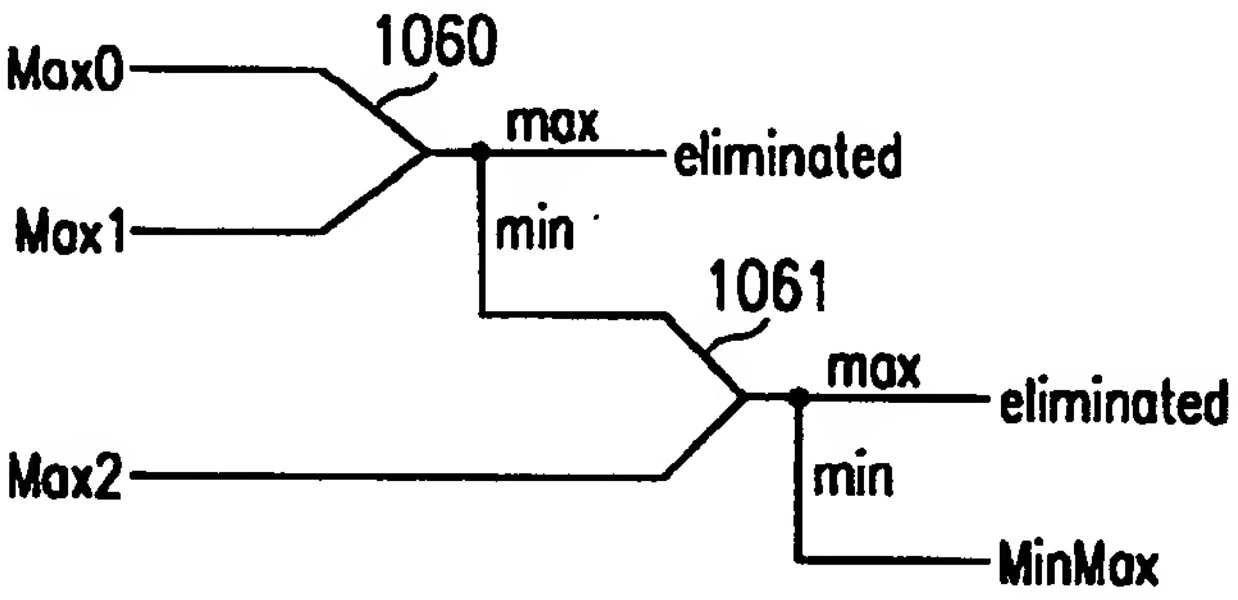


FIG. 48b

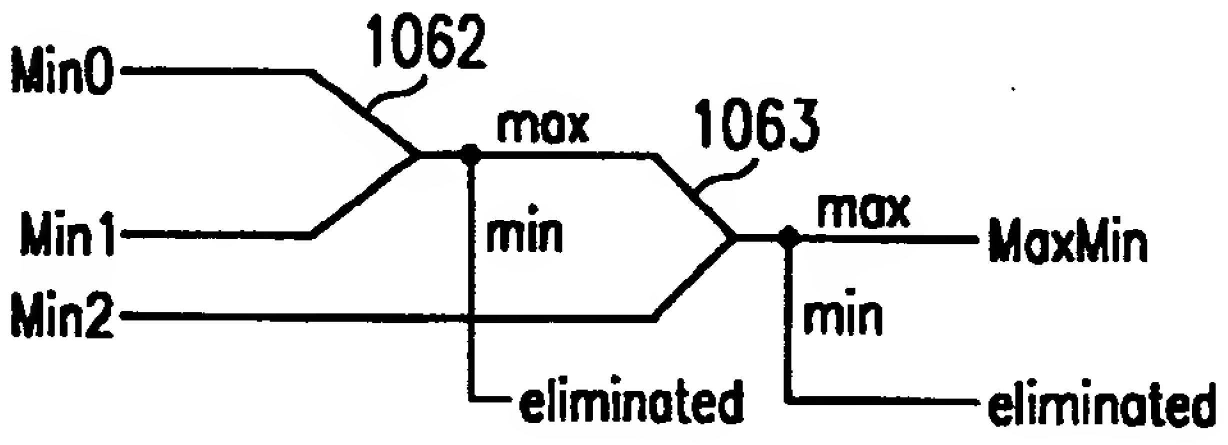


FIG. 48c

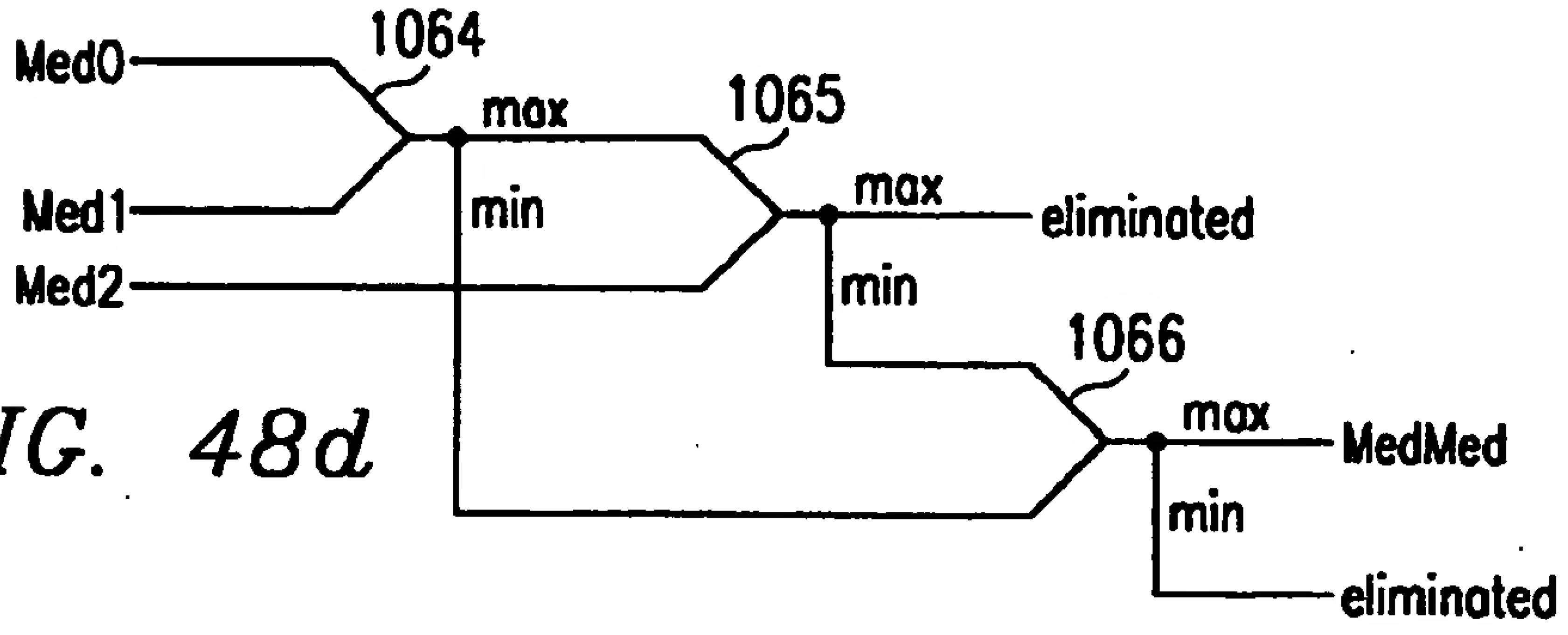


FIG. 48d

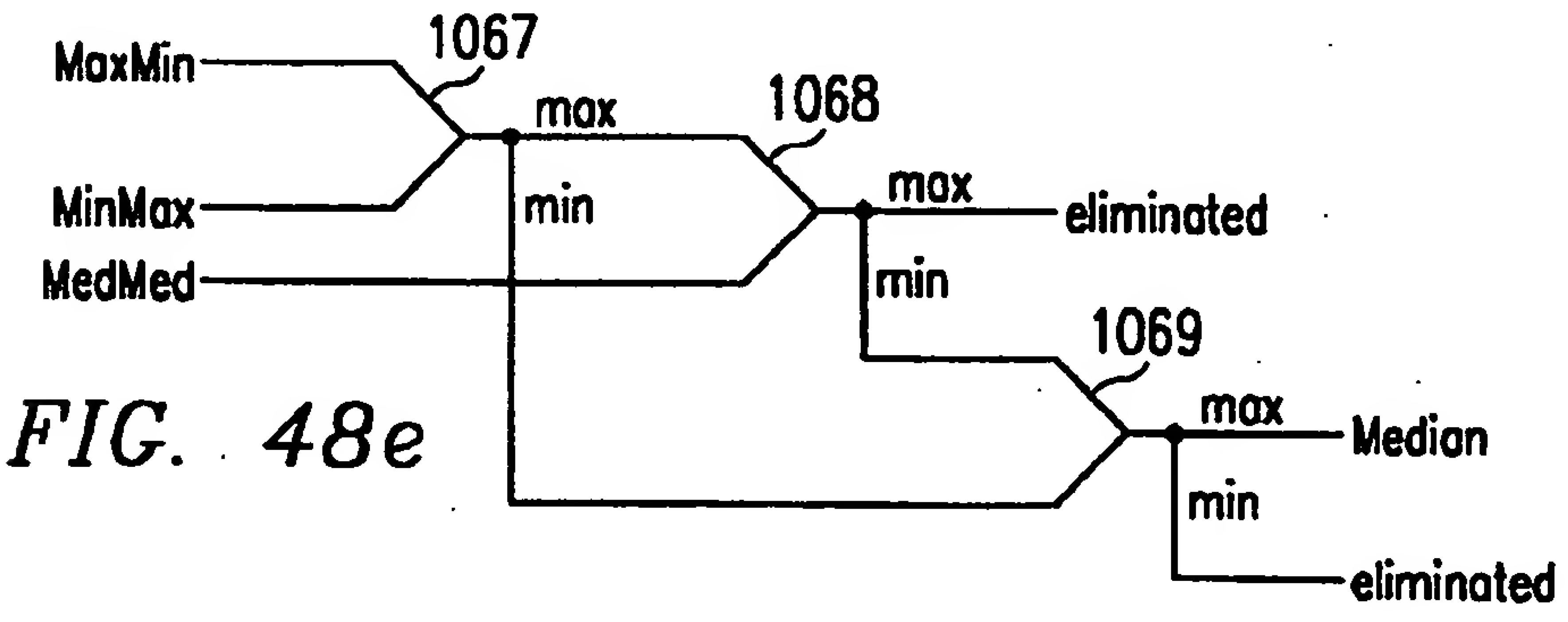


FIG. 48e

	Document	U	Title	Current OR
1	USD 6141751 A	□	User identifying method and system in computer communication network	713/170
2	US 5999198 A	□	Graphics address remapping table entry feature flags for customizing the operation of memory pages associated with an accelerated graphics port device	345/521
3	US 5999743 A	□	System and method for dynamically allocating accelerated graphics port memory space	710/56
4	US 5990914 A	□	Generating an error signal when accessing an invalid memory page	345/521
5	US 5936640 A	□	Accelerated graphics port memory mapped status and control registers	345/501
6	US 5933158 A	□	Use of a link bit to fetch entries of a graphic address remapping table	345/516
7	US 5926646 A	□	Context-dependent memory-mapped registers for transparent expansion of a register file	712/32
8	US 5923878 A	□	System, method and apparatus of directly executing an architecture-independent binary program	717/4
9	US 5917497 A	□	Method for maintaining contiguous texture memory for cache coherency	345/430
10	US 5914730 A	□	System and method for invalidating and updating individual GART table entries for accelerated graphics port transaction requests	345/521
11	US 5914727 A	□	Valid flag for disabling allocation of accelerated graphics port memory space	345/503
12	US 5898883 A	☒	Memory access mechanism for a parallel processing computer system with distributed shared memory	712/28
13	US 5864705 A	☒	Optimized environments for virtualizing physical subsystems independent of the operating system	712/32
14	US 5848423 A	☒	Garbage collection system and method for locating root set pointers in method activation records	707/206
15	US 5838987 A	☒	Processor for eliminating external isochronous subsystems	712/32
16	US 5819114 A	☒	Interruption recovery and resynchronization of events in a computer	710/57
17	US 5801720 A	☒	Data transfer from a graphics subsystem to system memory	345/526
18	US 5786825 A	☒	Virtual display subsystem in a computer	345/501
19	US 5781197 A	☒	Method for maintaining contiguous texture memory for cache coherency	345/430
20	US 5764999 A	☒	Enhanced system management mode with nesting	710/261
21	US 5748174 A	☒	Video display system including graphic layers with sizable, positionable windows and programmable priority	345/118

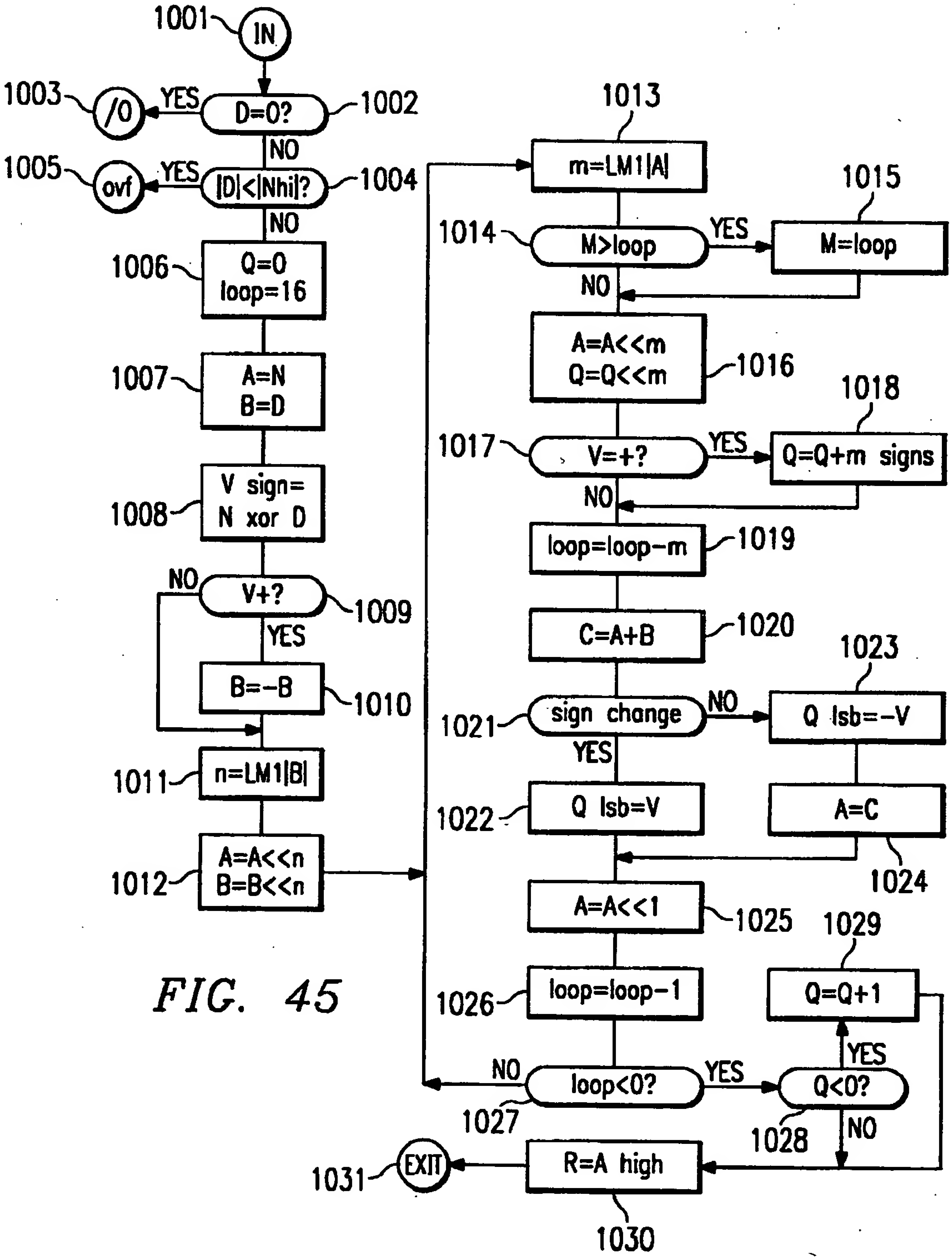

1001
IN
1003
/0
YES
D=0?
1002
NO
1005
ovf
YES
|D|<|Nhi|?
1004
NO
1006
Q=0
loop=16
1007
A=N
B=D
1008
V sign=
N xor D
NO
V+?
1009
YES
B=-B
1010
1011
n=LM1|B|
1012
A=A<<n
B=B<<n
1013
m=LM1|A|
1014
M>loop
YES
1015
M=loop
NO
A=A<<m
Q=Q<<m
1016
1017
V=+?
YES
1018
Q=Q+m signs
NO
loop=loop-m
1019
C=A+B
1020
1021
sign change
NO
1023
Q lsb=-V
YES
1022
Q lsb=V
A=C
1024
A=A<<1
1025
1029
1026
loop=loop-1
Q=Q+1
NO
loop<0?
YES
Q<0?
YES
1027
1028
NO
1031
EXIT
R=A high
1030

FIG. 45

	Document	U	Title	Current OR
22	USD 5706481 A	☒	Apparatus and method for integrating texture memory and interpolation logic in a computer system	345/519
23	US 5706461 A	☒	Method and apparatus for implementing virtual memory having multiple selected page sizes	711/203
24	US 5652857 A	☒	Disk control apparatus for recording and reproducing compression data to physical device of direct access type	711/113
25	US 5632028 A	☒	Hardware support for fast software emulation of unimplemented instructions	703/26
26	US 5570115 A	☒	Image processor	345/199
27	US 5555387 A	☒	Method and apparatus for implementing virtual memory having multiple selected page sizes	711/209
28	US 5548709 A	☒	Apparatus and method for integrating texture memory and interpolation logic in a computer system	345/510
29	US 5469431 A	☒	Method of and apparatus for channel mapping with relative service identification	370/254
30	US 5426727 A	☒	High-quality character generating system and method for use therein	345/467
31	US 5412766 A	☒	Data processing method and apparatus for converting color image data to non-linear palette	345/431
32	US 5367331 A	☒	Video codec, for a videophone terminal of an integrated services digital network	348/14
33	US 5287388 A	☒	Frequency offset removal method and apparatus	375/344
34	US 5278424 A	☒	Apparatus and method for correcting an offset value contained in an output of a turning angular velocity sensor	250/559.37
35	US 5198605 A	☒	Key touch data generation circuit of an electronic musical instrument	84/658
36	US 5166722 A	☒	Camera image shake detecting apparatus	396/54
37	US 5117493 A	☒	Pipelined register cache	711/140
38	US 5003524 A	☒	Optical disk drive with an accurately positioned objective lens	369/44.28
39	US 4965771 A	☒	Printer controller for connecting a printer to an information processor having a different protocol from that of a printer	358/1.13
40	US 4911550 A	☒	Optical type displacement measuring apparatus	356/376
41	US 4811206 A	☒	Data processing system with overlapped address translation and address computation	711/206
42	US 4374419 A	☒	Device for determining a radiation attenuation distribution in a plane of a body	378/11
43	US 4334155 A	☒	X-ray examination apparatus, comprising an examination table which can be swivelled around a horizontal axis	378/196

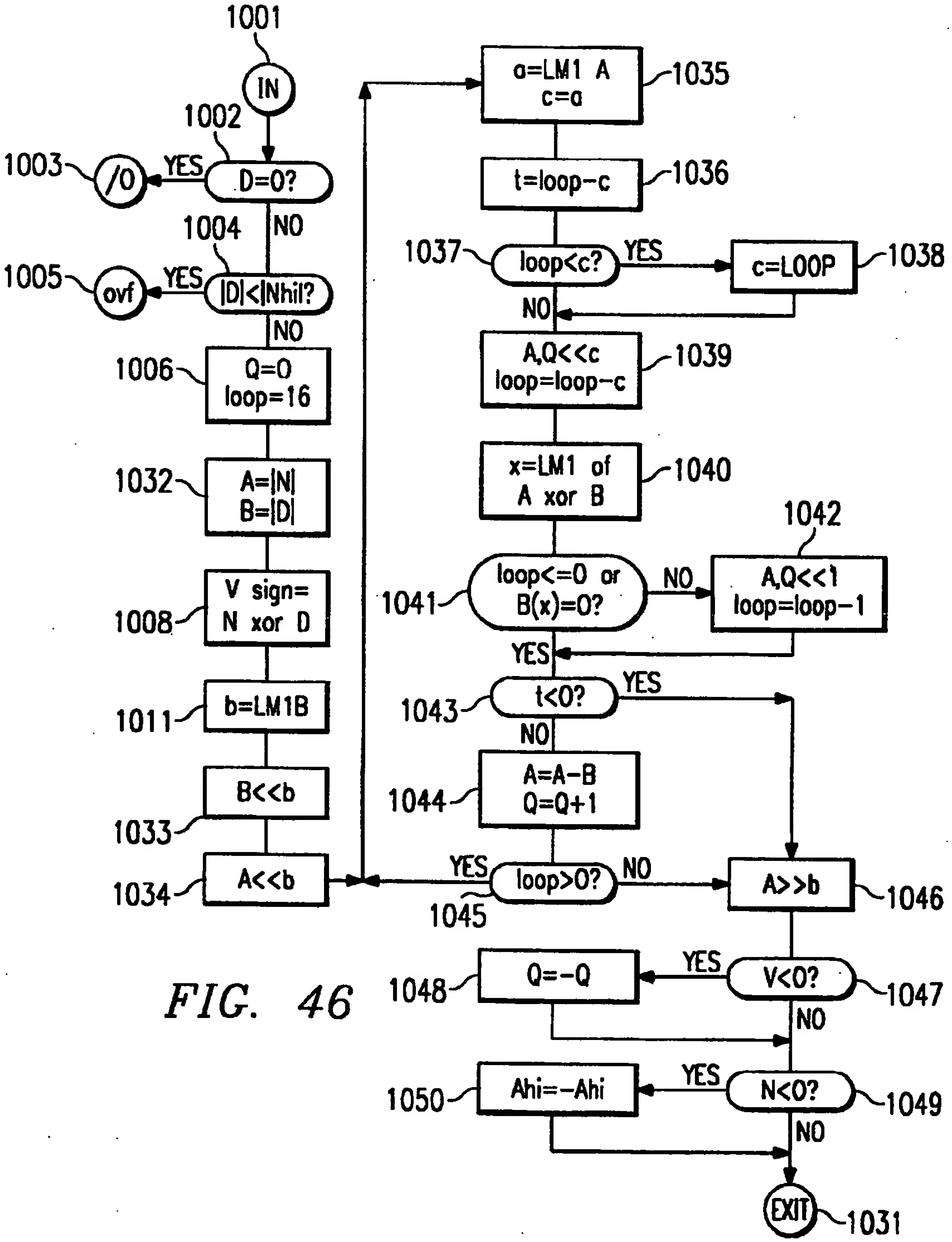

1001
IN
1002
1003
/0
YES
D=0?
NO
1004
1005
ovf
YES
|D|<|Nhi|?
NO
1006
Q=0
loop=16
1032
A=|N|
B=|D|
1008
V sign=
N xor D
1011
b=LM1B
B<<b
1033
1034
A<<b
a=LM1 A
c=a
1035
t=loop-c
1036
1037
loop<c?
YES
c=LOOP
1038
NO
A,Q<<c
loop=loop-c
1039
x=LM1 of
A xor B
1040
1042
1041
loop<=0 or
B(x)=0?
NO
A,Q<<1
loop=loop-1
YES
1043
t<0?
YES
NO
1044
A=A-B
Q=Q+1
YES
loop>0?
NO
1045
A>>b
1046
1048
Q=-Q
YES
V<0?
1047
NO
1050
Ahi=-Ahi
YES
N<0?
1049
NO
EXIT
1031

FIG. 46

	Document	U	Title	Current OR
1	USD 6061738 A	□	Method and system for accessing information on a network using message aliasing functions having shadow callback functions	709/245
2	US 6049667 A	□	Computer system, method of compiling and method of accessing address space with pointer of different width therefrom	717/5
3	US 5632028 A	□	Hardware support for fast software emulation of unimplemented instructions	703/26

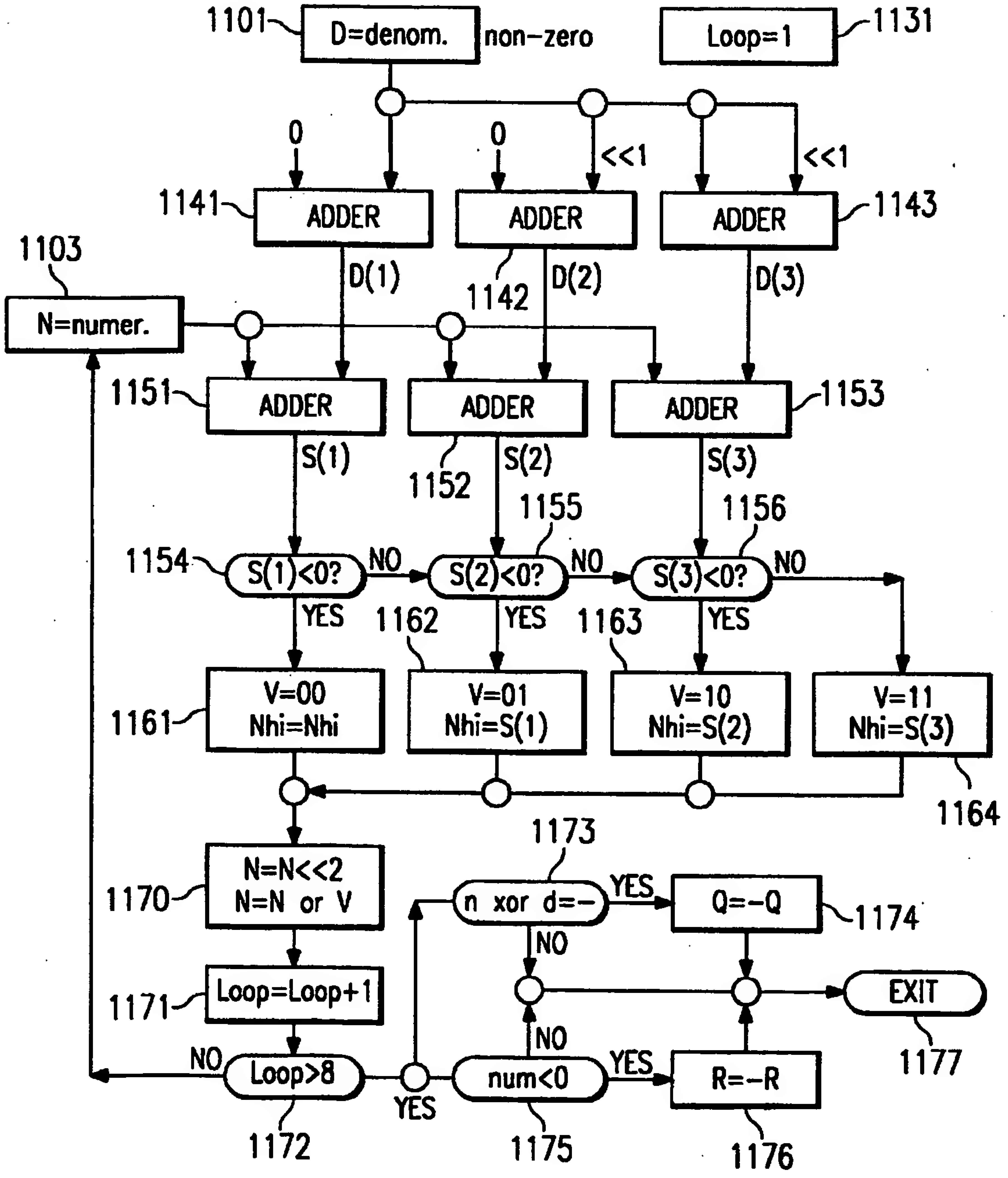

1101
D=denom.
non-zero
Loop=1
1131
0
0
<<1
<<1
1141
ADDER
ADDER
ADDER
1143
1103
D(1)
1142
D(2)
D(3)
N=numer.
1151
ADDER
ADDER
ADDER
1153
S(1)
1152
S(2)
S(3)
1155
1156
1154
S(1)<0?
NO
S(2)<0?
NO
S(3)<0?
NO
YES
1162
YES
1163
YES
1161
V=00
Nhi=Nhi
V=01
Nhi=S(1)
V=10
Nhi=S(2)
V=11
Nhi=S(3)
1164
1173
1170
N=N<<2
N=N or V
n xor d=−
YES
Q=−Q
1174
NO
1171
Loop=Loop+1
EXIT
NO
NO
Loop>8
num<0
YES
R=−R
YES
1177
1172
1175
1176

FIG. 51

	Document	U	Title	Current OR
1	USD 6064688 A	□	CDMA synchronous acquisition circuit	375/149
2	US 6061738 A	□	Method and system for accessing information on a network using message aliasing functions having shadow callback functions	709/245
3	US 6049667 A	□	Computer system, method of compiling and method of accessing address space with pointer of different width therefrom	717/5
4	US 5943691 A	□	Determination of array padding using collision vectors	711/172
5	US 5918991 A	□	Printing apparatus and control method for pitch and motion commands	400/303
6	US 5764094 A	□	Level shift circuit for analog signal and signal waveform generator including the same	327/333
7	US 5632028 A	□	Hardware support for fast software emulation of unimplemented instructions	703/26
8	US 5526502 A	□	Memory interface	711/202
9	US 5224178 A	□	Extending dynamic range of stored image database	382/166
10	US 4965867 A	□	Offset compensation circuit	341/118
11	US 4833604 A	□	Method for the relocation of linked control blocks	707/200
12	US 4366536 A	□	Modular digital computer system for storing and selecting data processing procedures and data	711/206
13	US 4315252 A	□	Apparatus for detecting the relative position of two movable bodies	341/1
14	US 3894347 A	□	Three dimensional chaff simulation system	434/5

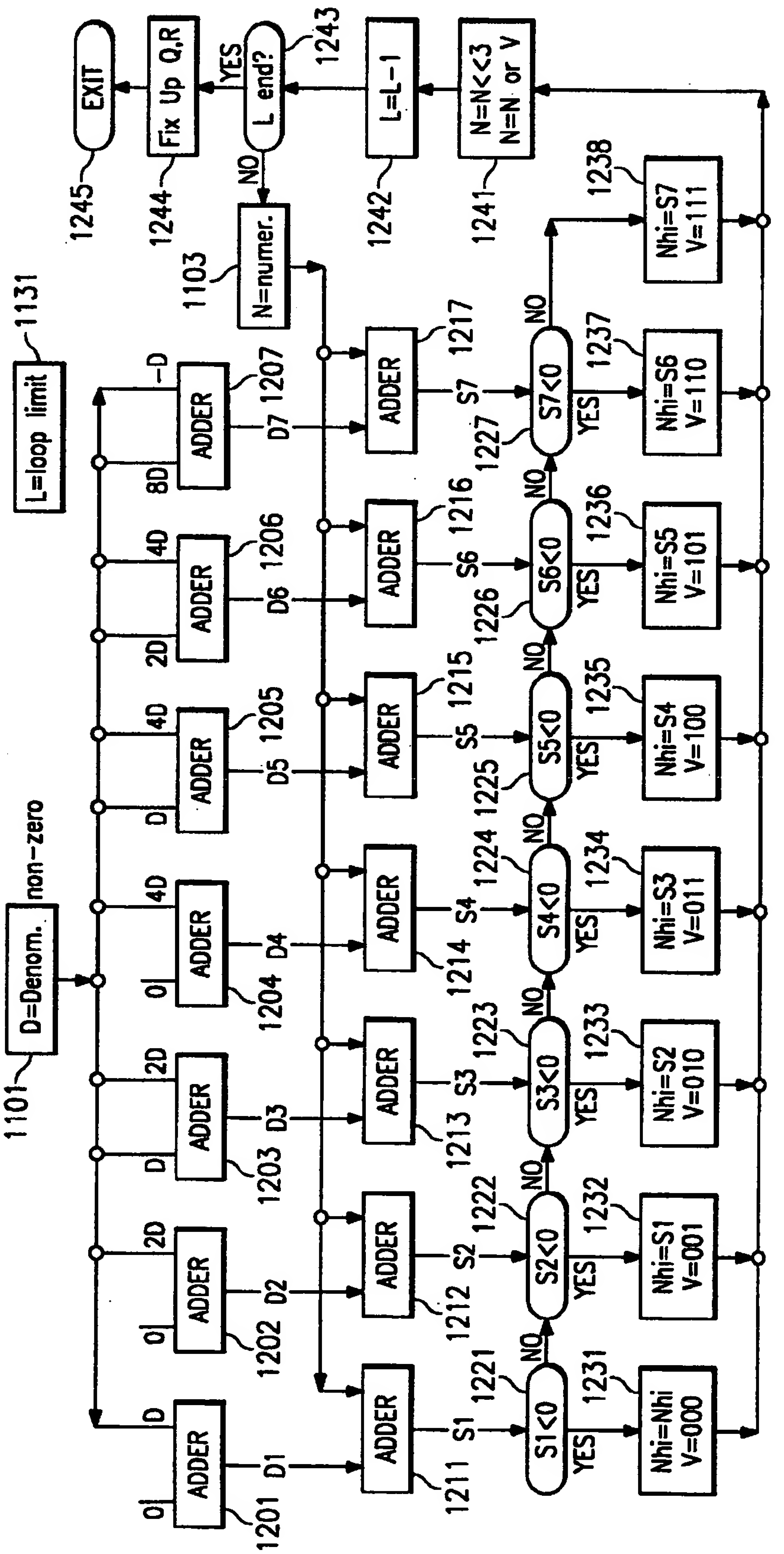

L=loop limit
1131
D=Denom.
non-zero
1101
ADDER
1201
0
D
D1
ADDER
1202
0
2D
D2
ADDER
1203
D
2D
D3
ADDER
1204
0
4D
D4
ADDER
1205
D
4D
D5
ADDER
1206
2D
4D
D6
ADDER
1207
8D
-D
D7
N=numer.
1103
ADDER
1211
S1
ADDER
1212
S2
ADDER
1213
S3
ADDER
1214
S4
ADDER
1215
S5
ADDER
1216
S6
ADDER
1217
S7
S1<0
1221
S2<0
1222
S3<0
1223
S4<0
1224
S5<0
1225
S6<0
1226
S7<0
1227
YES
NO
Nhi=Nhi
V=000
1231
Nhi=S1
V=001
1232
Nhi=S2
V=010
1233
Nhi=S3
V=011
1234
Nhi=S4
V=100
1235
Nhi=S5
V=101
1236
Nhi=S6
V=110
1237
Nhi=S7
V=111
1238
N=N<<3
N=N or V
1241
L=L-1
1242
L end?
1243
Fix Up Q,R
1244
EXIT
1245

FIG. 53

	Document	U	Title	Current OR
1	USD 6101592 A	□	Methods and apparatus for scalable instruction set architecture with dynamic compact instructions	712/20
2	US 6075470 A	☒	Block-wise adaptive statistical data compressor	341/107
3	US 6044450 A	☒	Processor for VLIW instruction	712/24
4	US 6035123 A	☒	Determining hardware complexity of software operations	717/9
5	US 5983004 A	☒	Computer, memory, telephone, communications, and transportation system and methods	709/227 ;
6	US 5964861 A	☒	Method for writing a program to control processors using any instructions selected from original instructions and defining the instructions used as a new instruction set	712/23
7	US 5950440 A	☒	Vehicle air conditioner with compressor noise reduction control	62/133
8	US 5935256 A	☒	Parallel processing integrated circuit tester	713/400
9	US 5931953 A	☒	Parallel processing integrated circuit tester	713/500
10	US 5931952 A	☒	Parallel processing integrated circuit tester	713/400
11	US 5930508 A	☒	Method for storing and decoding instructions for a microprocessor having a plurality of function units	717/6
12	US 5909587 A	☒	Multi-chip superscalar microprocessor module	712/1
13	US 5899961 A	☒	Electronic circuit or board tester with compressed data-sequences	702/117
14	US 5881260 A	☒	Method and apparatus for sequencing and decoding variable length instructions with an instruction boundary marker within each instruction	712/210
15	US 5878267 A	☒	Compressed instruction format for use in a VLIW processor and processor for processing such instructions	712/24
16	US 5867712 A	☒	Single chip integrated circuit system architecture for document instruction set computing	717/4 ;
17	US 5852741 A	☒	VLIW processor which processes compressed instruction format	712/24
18	US 5832289 A	☒	System for estimating worst time duration required to execute procedure calls and looking ahead/preparing for the next stack operation of the forthcoming procedure calls	712/30
19	US 5822578 A	☒	System for inserting instructions into processor instruction stream in order to perform interrupt processing	712/244
20	US 5819308 A	☒	Method for buffering and issuing instructions for use in high-performance superscalar microprocessors	711/108

ARITHMETIC LOGIC UNIT HAVING PLURAL INDEPENDENT SECTIONS AND REGISTER STORING RESULTANT INDICATOR BIT FROM EVERY SECTION

CROSS REFERENCE TO RELATED APPLICATIONS

This application relates to improvements in the inventions disclosed in the following copending U.S. patent applications, all of which are assigned to Texas Instruments:

U.S. patent application Ser. No. 08/263,501, filed Jun. 21, 1994 entitled "MULTI-PROCESSOR WITH CROSSBAR LINK OF PROCESSORS AND MEMORIES AND METHOD OF OPERATION", a continuation of U.S. patent application Ser. No. 08/135,754, filed Oct. 12, 1993, and now abandoned, a continuation of U.S. patent application Ser. No. 07/933,865, filed Aug. 21, 1993, and now abandoned, a continuation of U.S. patent application Ser. No. 435,591 filed Nov. 17, 1989 and now abandoned;

U.S. Pat. No. 5,212,777, issued May 18, 1993, filed Nov. 17, 1989 and entitled "SIMD/MIMD RECONFIGURABLE MULTI-PROCESSOR AND METHOD OF OPERATION";

U.S. patent application Ser. No. 08/264,111 filed Jun. 22, 1994, entitled "RECONFIGURABLE COMMUNICATIONS FOR MULTI-PROCESSOR AND METHOD OF OPERATION," a continuation of U.S. patent application Ser. No. 07/895,565, filed Jun. 5, 1992, and now abandoned, a continuation of U.S. patent application Ser. No. 07/437,856, filed Nov. 17, 1989 and now abandoned;

U.S. patent application Ser. No. 08/264,582, filed Jun. 22, 1994, entitled "REDUCED AREA OF CROSSBAR AND METHOD OF OPERATION", a continuation of U.S. patent application Ser. No. 07/437,852, filed Nov. 17, 1989, and now abandoned;

U.S. patent application Ser. No. 08/032,530 filed Mar. 15, 1993 entitled "SYNCHRONIZED MIMD MULTIPROCESSING SYSTEM AND METHOD OF OPERATION," a continuation of U.S. patent application Ser. No. 07/437,853 filed Nov. 17, 1989 and now abandoned;

U.S. Pat. No. 5,197,140 issued Mar. 23, 1993 filed Nov. 17, 1989 and entitled "SLICED ADDRESSING MULTIPROCESSOR AND METHOD OF OPERATION";

U.S. Pat. No. 5,339,447, issued Aug. 16, 1994, filed Nov. 17, 1989 entitled "ONES COUNTING CIRCUIT, UTILIZING A MATRIX OF INTERCONNECTED HALF-ADDERS, FOR COUNTING THE NUMBER OF ONES IN A BINARY STRING OF IMAGE DATA;

U.S. Pat. No. 5,239,654 issued Aug. 24, 1993 filed Nov. 17, 1989 and entitled "DUAL MODE SIMD/MIND PROCESSOR PROVIDING REUSE OF MIMD INSTRUCTION MEMORIES AS DATA MEMORIES WHEN OPERATING IN SAID MODE";

U.S. Pat. No. 5,410,649, filed Jun. 29, 1992 entitled "IMAGING COMPUTER AND METHOD OF OPERATION", a continuation of U.S. patent application Ser. No. 07/437,854, filed Nov. 17, 1989 and now abandoned; and

U.S. Pat. No. 5,226,125 issued Jul. 6, 1993 filed Nov. 17, 1989 and entitled "SWITCH MATRIX HAVING INTEGRATED CROSSPOINT LOGIC AND METHOD OF OPERATION".

This application is also related to the following concurrently filed U.S. patent applications, which include the same disclosure:

U.S. Pat. No. 5,490,828, "THREE INPUT ARITHMETIC LOGIC UNIT WITH BARREL ROTATOR";

U.S. patent application Ser. No. 08/160,118, "MEMORY STORE FROM A REGISTER PAIR CONDITIONAL" and now pending;

U.S. Pat. No. 5,442,581, "ITERATIVE DIVISION APPARATUS, SYSTEM AND METHOD FORMING PLURAL QUOTIENT BITS PER ITERATION", a continuation of U.S. patent application Ser. No. 08/160,115, concurrently filed with this application and now abandoned;

U.S. patent application Ser. No. 08/158,285, "THREE INPUT ARITHMETIC LOGIC UNIT FORMING MIXED ARITHMETIC AND BOOLEAN COMBINATIONS", and now pending;

U.S. patent application Ser. No. 08/160,119, "METHOD, APPARATUS AND SYSTEM FORMING THE SUM OF DATA IN PLURAL EQUAL SECTIONS OF A SINGLE DATA WORD", and now pending;

U.S. Pat. No. 5,512,896, "HUFFMAN ENCODING METHOD, CIRCUITS AND SYSTEM EMPLOYING MOST SIGNIFICANT BIT CHANGE FOR SIZE DETECTION";

U.S. Pat. No. 5,479,166, "HUFFMAN DECODING METHOD, CIRCUIT AND SYSTEM EMPLOYING CONDITIONAL SUBTRACTION FOR CONVERSION OF NEGATIVE NUMBERS";

U.S. patent application Ser. No. 08/160,112, "METHOD, APPARATUS AND SYSTEM FOR SUM OF PLURAL ABSOLUTE DIFFERENCES", and now pending;

U.S. patent application Ser. No. 08/160,120, "ITERATIVE DIVISION APPARATUS, SYSTEM AND METHOD EMPLOYING LEFT MOST ONE'S DETECTION AND LEFT MOST ONE'S DETECTION WITH EXCLUSIVE OR", and now pending;

U.S. patent application Ser. No. 08/160,114, "ADDRESS GENERATOR EMPLOYING SELECTIVE MERGE OF TWO INDEPENDENT ADDRESSES", and now pending;

U.S. Pat. No. 5,420,809, "METHOD, APPARATUS AND SYSTEM METHOD FOR CORRELATION";

U.S. Pat. No. 5,509,129, "LONG INSTRUCTION WORD CONTROLLING PLURAL INDEPENDENT PROCESSOR OPERATIONS";

U.S. patent application Ser. No. 08/159,346, "ROTATION REGISTER FOR ORTHOGONAL DATA TRANSFORMATION"; and now pending;

U.S. patent application Ser. No. 08/159,652, "MEDIAN FILTER METHOD, CIRCUIT AND SYSTEM", and now pending;

U.S. patent application Ser. No. 08/159,344, "ARITHMETIC LOGIC UNIT WITH CONDITIONAL REGISTER SOURCE SELECTION and now pending;

U.S. patent application Ser. No. 08/160,301, "APPARATUS, SYSTEM AND METHOD FOR DIVISION BY ITERATION", and now pending;

U.S. patent application Ser. No. 08/159,650, "MULTIPLY ROUNDING USING REDUNDANT CODED MULTIPLY RESULT", and now pending;

U.S. Pat. No. 5,446,651, "SPLIT MULTIPLY OPERATION";

U.S. patent application Ser. No. 08,482,697, filed Jun. 7, 1995, "MIXED CONDITION TEST CONDITIONAL AND BRANCH OPERATIONS INCLUDING CONDITIONAL TEST FOR ZERO", a continuation of U.S. patent application Ser. No. 08/158,741, concurrently filed with this application and now abandoned;

	Document	U	Title	Current OR
21	US 5819058 A	☒	Instruction compression and decompression system and method for a processor	712/210
22	US 5806068 A	☒	Document data processor for an object-oriented knowledge management system containing a personal database in communication with a packet processor	707/103
23	US 5784585 A	☒	Computer system for executing instruction stream containing mixed compressed and uncompressed instructions by automatically detecting and expanding compressed instructions	712/209
24	US 5748642 A	☒	Parallel processing integrated circuit tester	714/724
25	US 5745758 A	☒	System for regulating multicomputer data transfer by allocating time slot to designated processing task according to communication bandwidth capabilities and modifying time slots when bandwidth change	709/102
26	US 5734854 A	☒	Fast instruction decoding in a pipeline processor	712/205
27	US 5659785 A	☒	Array processor communication architecture with broadcast processor instructions	712/11
28	US 5630085 A	☒	Microprocessor with improved instruction cycle using time-compressed fetching	712/207
29	US 5621907 A	☒	Microprocessor with memory storing instructions for time-compressed fetching of instruction data for a second cycle within a first machine cycle	712/207
30	US 5600844 A	☒	Single chip integrated circuit system architecture for document installation set computing	345/507
31	US 5592635 A	☒	Technique for accelerating instruction decoding of instruction sets with variable length opcodes in a pipeline microprocessor	712/210
32	US 5577259 A	☒	Instruction processor control system using separate hardware and microcode control signals to control the pipelined execution of multiple classes of machine instructions	712/41
33	US 5548766 A	☒	Microprocessor for selecting data bus terminal width regardless of data transfer mode	710/127
34	US 5548544 A	☒	Method and apparatus for rounding the result of an arithmetic operation	708/497
35	US 5510857 A	☒	Motion estimation coprocessor	348/699
36	US 5491811 A	☒	Cache system using mask bits to recorder the sequences for transfers of data through cache to system memory	711/144
37	US 5490259 A	☒	Logical-to-real address translation based on selective use of first and second TLBs	711/202
38	US 5448310 A	☒	Motion estimation coprocessor	348/699

U.S. patent application Ser. No. 08/160,302, "PACKED WORD PAIR MULTIPLY OPERATION", and now abandoned;

U.S. patent application Ser. No. 08/160,573, "THREE INPUT ARITHMETIC LOGIC UNIT WITH SHIFTER", and now pending;

U.S. patent application Ser. No. 08/159,282, "THREE INPUT ARITHMETIC LOGIC UNIT WITH MASK GENERATOR", and now pending;

U.S. patent application Ser. No. 08/160,111, "THREE INPUT ARITHMETIC LOGIC UNIT WITH BARREL ROTATOR AND MASK GENERATOR", and now pending;

U.S. patent application Ser. No. 08/160,298, "THREE INPUT ARITHMETIC LOGIC UNIT WITH SHIFTER AND MASK GENERATOR", and now pending;

U.S. Pat. No. 5,485,411, "THREE INPUT ARITHMETIC LOGIC UNIT FORMING THE SUM OF A FIRST INPUT ADDED WITH A FIRST BOOLEAN COMBINATION OF A SECOND INPUT AND THIRD INPUT PLUS A SECOND BOOLEAN COMBINATION OF THE SECOND AND THIRD INPUTS";

U.S. Pat. No. 5,465,224, "THREE INPUT ARITHMETIC LOGIC UNIT FORMING THE SUM OF FIRST BOOLEAN COMBINATION OF FIRST, SECOND AND THIRD INPUTS PLUS A SECOND BOOLEAN COMBINATION OF FIRST, SECOND AND THIRD INPUTS";

U.S. Pat. No. 5,493,524, "THREE INPUT ARITHMETIC LOGIC UNIT EMPLOYING CARRY PROPAGATE LOGIC", a continuation of U.S. patent application Ser. No. 08/159,640, filed concurrently with this application and now abandoned; and

U.S. patent application Ser. No. 08/160,300, "DATA PROCESSING APPARATUS, SYSTEM AND METHOD FOR IF, THEN, ELSE OPERATION USING WRITE PRIORITY", and now pending.

TECHNICAL FIELD OF THE INVENTION

The technical field of this invention is the field of digital data processing and more particularly microprocessor circuits, architectures and methods for digital data processing especially digital image/graphics processing.

BACKGROUND OF THE INVENTION

This invention relates to the field of computer graphics and in particular to bit mapped graphics. In bit mapped graphics computer memory stores data for each individual picture element or pixel of an image at memory locations that correspond to the location of that pixel within the image. This image may be an image to be displayed or a captured image to be manipulated, stored, displayed or retransmitted. The field of bit mapped computer graphics has benefited greatly from the lowered cost and increased capacity of dynamic random access memory (DRAM) and the lowered cost and increased processing power of microprocessors. These advantageous changes in the cost and performance of component parts enable larger and more complex computer image systems to be economically feasible.

The field of bit mapped graphics has undergone several stages in evolution of the types of processing used for image data manipulation. Initially a computer system supporting bit mapped graphics employed the system processor for all bit mapped operations. This type of system suffered several drawbacks. First, the computer system processor was not particularly designed for handling bit mapped graphics. Design choices that are very reasonable for general purpose computing are unsuitable for bit mapped graphics systems. Consequently some routine graphics tasks operated slowly. In addition, it was quickly discovered that the processing needed for image manipulation of bit mapped graphics was so loading the computational capacity of the system processor that other operations were also slowed.

The next step in the evolution of bit mapped graphics processing was dedicated hardware graphics controllers. These devices can draw simple figures, such as lines, ellipses and circles, under the control of the system processor. Many of these devices can also do pixel block transfers (PixBlt). A pixel block transfer is a memory move operation of image data from one portion of memory to another. A pixel block transfer is useful for rendering standard image elements, such as alphanumeric characters in a particular type font, within a display by transfer from nondisplayed memory to bit mapped display memory. This function can also be used for tiling by transferring the same small image to the whole of bit mapped display memory. The built-in algorithms for performing some of the most frequently used graphics functions provide a way of improving system performance. However, a useful graphics computer system often requires many functions besides those few that are implemented in such a hardware graphics controller. These additional functions must be implemented in software by the system processor. Typically these hardware graphics controllers allow the system processor only limited access to the bit map memory, thereby limiting the degree to which system software can augment the fixed set of functions of the hardware graphics controller.

The graphics system processor represents yet a further step in the evolution of bit mapped graphics processing. A graphics system processor is a programmable device that has all the attributes of a microprocessor and also includes special functions for bit mapped graphics. The TMS34010 and TMS34020 graphics system processors manufactured by Texas Instruments Incorporated represent this class of devices. These graphics system processors respond to a stored program in the same manner as a microprocessor and include the capability of data manipulation via an arithmetic logic unit, data storage in register files and control of both program flow and external data memory. In addition, these devices include special purpose graphics manipulation hardware that operate under program control. Additional instructions within the instruction set of these graphics system processors controls the special purpose graphics hardware. These instructions and the hardware that supports them are selected to perform base level graphics functions that are useful in many contexts. Thus a graphics system processor can be programmed for many differing graphics applications using algorithms selected for the particular problem. This provides an increase in usefulness similar to that provided by changing from hardware controllers to programmed microprocessors. Because such graphics system processors are programmable devices in the same manner as microprocessors, they can operate as stand alone graphics processors, graphics co-processors slaved to a system processor or tightly coupled graphics controllers.

New applications are driving the desire to provide more powerful graphics functions. Several fields require more cost effective graphics operations to be economically feasible. These include video conferencing, multi-media computing with full motion video, high definition television, color facsimile and digital photography. Each of these fields presents unique problems, but image data compression and decompression are common themes. The amount of transmission bandwidth and the amount of storage capacity

	Document ID	U	Title	Current OR
39	US 5415140 A	☒	Method for moving a group of members along a trajectory by moving a second group of members with a reciprocating motion along another trajectory	123/197.1
40	US 5371860 A	☒	Programmable controller	710/22
41	US 5349667 A	☒	Interrupt control system for microprocessor for handling a plurality of maskable interrupt requests	710/267
42	US 5323618 A	☒	Heat storage type air conditioning apparatus	62/149
43	US 5283891 A	☒	Error information saving apparatus of computer	714/45
44	US 5247627 A	☒	Digital signal processor with conditional branch decision unit and storage of conditional branch decision results	712/236
45	US 5237667 A	☒	Digital signal processor system having host processor for writing instructions into internal processor memory	712/248
46	US 5222241 A	☒	Digital signal processor having duplex working registers for switching to standby state during interrupt processing	712/228
47	US 5206940 A	☒	Address control and generating system for digital signal-processor	711/218
48	US 5045993 A	☒	Digital signal processor	712/236
49	US 5031096 A	☒	Method and apparatus for compressing the execution time of an instruction stream executing in a pipelined processor	711/169
50	US 5019967 A	☒	Pipeline bubble compression in a computer system	712/219
51	US 5008807 A	☒	Data processing apparatus with abbreviated jump field	712/213
52	US 4881168 A	☒	Vector processor with vector data compression/expansion capability	712/5
53	US 4833599 A	☒	Hierarchical priority branch handling for parallel execution in a parallel processor	712/236
54	US 4770602 A	☒	Method of capacity controlling of multistage compressor and apparatus therefor	415/29
55	US 4706466 A	☒	Under the counter ice making machine	62/138
56	US 4654484 A	☒	Video compression/expansion system	348/17
57	US 4532589 A	☒	Digital data processor with two operation units	712/217
58	US 4484452 A	☒	Heat pump refrigerant charge control system	62/174
59	US 4481784 A	☒	Automotive air conditioning compressor control system	62/133

US005826054A

United States Patent [19]

Jacobs et al.

[11] **Patent Number: 5,826,054**

[45] **Date of Patent: Oct. 20, 1998**

[54] **COMPRESSED INSTRUCTION FORMAT FOR USE IN A VLIW PROCESSOR**

[75] Inventors: **Eino Jacobs**, Palo Alto; **Michael Ang**, Santa Clara, both of Calif.

[73] Assignee: **Philips Electronics North America Corporation**, N.Y., N.Y.

[21] Appl. No.: **649,733**

[22] Filed: **May 15, 1996**

[51] **Int. Cl.**^{6} **G06F 12/04**; G06F 9/22; H03M 7/30

[52] **U.S. Cl.** **395/389**; 358/432; 364/725.03

[58] **Field of Search** .... 395/24, 800.23, 395/417, 287, 583, 485, 582, 382, 500, 389, 800.24, 706; 341/55; 364/DIG. 1, DIG. 2, 725.03; 365/201; 711/1; 382/236; 358/432, 427, 433, 426

[56] **References Cited**

U.S. PATENT DOCUMENTS

4,437,149	3/1984	Pomerene et al.	395/389
5,057,837	10/1991	Colwell et al.	341/55
5,179,680	1/1993	Colwell et al.	711/125
5,265,258	11/1993	Fiene et al.	395/485
5,381,531	1/1995	Hanawa et al.	395/582
5,471,593	11/1995	Branigin	395/24
5,530,817	6/1996	Masubuchi	395/800.24
5,652,852	7/1997	Yokota	395/384
5,669,001	9/1997	Moreno	395/706

FOREIGN PATENT DOCUMENTS

WO9519006	7/1995	WIPO	G06F 15/78

OTHER PUBLICATIONS

Philips hopes to displace DSPs with VLIW: TriMedia processors aimed at future multimedia embedded apps. Brian Case, Microprocessor Report vol. 8, nr 16, p. 12(4) Dec. 5, 1994.

Wang et al., "The Feasibility of Using Compression to Increase Memory System Performance", Proc. 2nd Int. Workshop on Modeling Analysis, and Simulation of Computer and Telecommunications Systems, pp. 107–113.

Schroder et al., "Program Compression on the Instruction Systolic Array", Parallel Computing, vol. 17, 1991, No. 2–3, pp. 207–219.

Wolfe et al., "Executing Compressed Programs on An Embedded RISC Architecture", J. Computer and Software Engineering, vol. 2, No. 3, pp. 315–327, (1994).

Kozuch et al., "Compression of Embedded System Programs", Proc. 1994 IEEE Inter. Conf. on computer Design: VLSI in Computers and Processors (Oct. 10–12, 1994, Cambridge, MA,) pp. 270–277.

Russ, *An Information–Theoretic Approach To Analysis of Computer Archetectures and Compression of Instruction Memory Usage*, UMI Dissertation Services, Entire Book.

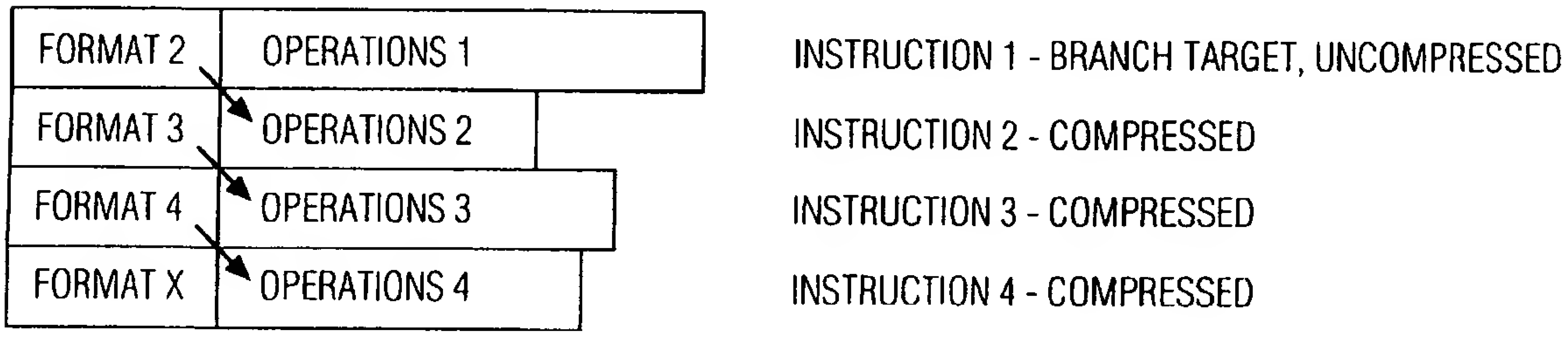
Primary Examiner—Daniel H. Pan
Attorney, Agent, or Firm—Anne E. Barschall

[57] **ABSTRACT**

A compressed instruction format for a VLIW processor allows greater efficiency in use of cache and memory. Instructions are byte aligned and variable length. Branch targets are uncompressed. Format bits specify how many issue slots are used in a following instruction. NOPS are not stored in memory. Individual operations are compressed according to features such as whether they are resultless, guarded, short, zeroary, unary, or binary. Instructions are stored in compressed form in memory and in cache. Instructions are decompressed on the fly after being read out from cache.

20 Claims, 15 Drawing Sheets

Microfiche Appendix Included
(2 Microfiche, 66 Pages)


FORMAT 2
OPERATIONS 1
INSTRUCTION 1 - BRANCH TARGET, UNCOMPRESSED
FORMAT 3
OPERATIONS 2
INSTRUCTION 2 - COMPRESSED
FORMAT 4
OPERATIONS 3
INSTRUCTION 3 - COMPRESSED
FORMAT X
OPERATIONS 4
INSTRUCTION 4 - COMPRESSED

	Document	U	Title	Current OR
60	USD 4449489 A	☒	Varying geometric compression ratio engine	123/48R
61	US 4297852 A	☒	Refrigerator defrost control with control of time interval between defrost cycles	62/153
62	US 4288816 A	☒	Compressed image producing system	382/232
63	US 4221176 A	☒	Profile stitching apparatus and method	112/102.5
64	US 4058850 A	☒	Programmable controller	711/117
65	US 3936182 A	☒	Control arrangement for an electrostatographic reproduction apparatus	399/77
66	US 3885207 A	☒	Optimized editing system for a servo controlled program recording system	318/568.14
67	US 3812475 A	☒	DATA SYNCHRONIZER	710/7
68	US 3717850 A	☒	PROGRAMMED DATA PROCESSING WITH FACILITATED TRANSFERS	712/205;

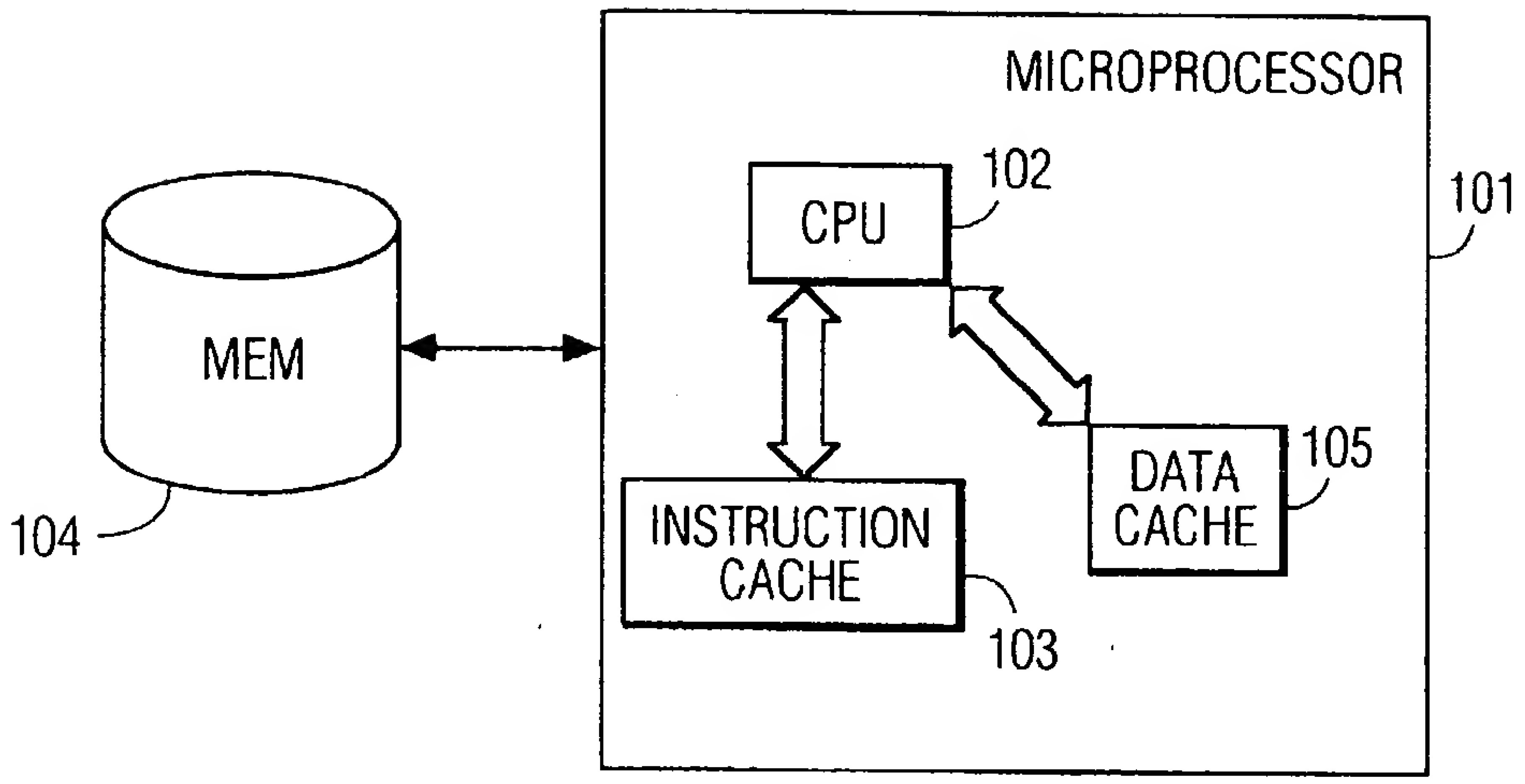

MICROPROCESSOR
101
CPU
102
MEM
104
DATA
CACHE
105
INSTRUCTION
CACHE
103

FIG. 1a

	Document	U	Title	Current OR
1	USD 6128721 A	□	Temporary pipeline register file for a superpipelined superscalar processor	712/23
2	US 6116768 A	☒	Three input arithmetic logic unit with barrel rotator	708/236
3	US 6112289 A	☒	Data processor	712/23
4	US 6105127 A	☒	Multithreaded processor for processing multiple instruction streams independently of each other by flexibly controlling throughput in each instruction stream	712/215
5	US 6098163 A	☒	Three input arithmetic logic unit with shifter	712/20
6	US 6092177 A	☒	Computer architecture capable of execution of general purpose multiple instructions	712/23
7	US 6070003 A	☒	System and method of memory access in apparatus having plural processors and plural memories	710/132
8	US 6067613 A	☒	Rotation register for orthogonal data transformation	712/32
9	US 6058473 A	☒	Memory store from a register pair conditional upon a selected status bit	712/225
10	US 6038584 A	☒	Synchronized MIMD multi-processing system and method of operation	709/248
11	US 6032170 A	☒	Long instruction word controlling plural independent processor operations	708/620
12	US 6026484 A	☒	Data processing apparatus, system and method for if, then, else operation using write priority	712/226
13	US 6016538 A	☒	Method, apparatus and system forming the sum of data in plural equal sections of a single data word	712/32
14	US 6009506 A	☒	Computer architecture capable of concurrent issuance and execution of general purpose multiple instructions	712/23
15	US 5995747 A	☒	Three input arithmetic logic unit capable of performing all possible three operand boolean operations with shifter and/or mask generator	712/221
16	US 5995748 A	☒	Three input arithmetic logic unit with shifter and/or mask generator	712/221
17	US 5991902 A	☒	Memory apparatus and data processor using the same	714/710
18	US 5974539 A	☒	Three input arithmetic logic unit with shifter and mask generator	712/221
19	US 5961635 A	☒	Three input arithmetic logic unit with barrel rotator and mask generator	712/221
20	US 5960193 A	☒	Apparatus and system for sum of plural absolute differences	712/221
21	US 5954435 A	☒	Memory apparatus and data processor using the same	714/42

US005463746A

United States Patent [19]

Brodnax et al.

[11] **Patent Number: 5,463,746**

[45] **Date of Patent: Oct. 31, 1995**

[54] **DATA PROCESSING SYSTEM HAVING PREDICTION BY USING AN EMBEDDED GUESS BIT OF REMAPPED AND COMPRESSED OPCODES**

[75] Inventors: **Timothy B. Brodnax**, Austin, Tex.; **Bryan K. Bullis**, Woodbridge; **Steven A. King**, Herndon, both of Va.; **Peter M. Kogge**, Endicott, N.Y.; **Dale A. Rickard**, Manassas, Va.

[73] Assignee: **International Business Machines Corp.**, Armonk, N.Y.

[21] Appl. No.: **968,790**

[22] Filed: **Oct. 30, 1992**

[51] **Int. Cl.^{6}** **G06F 9/38**

[52] **U.S. Cl.** **395/375**; 364/DIG. 1; 364/263.1; 364/261.7; 364/260.6; 364/260.7

[58] **Field of Search** 395/375, 800

[56] **References Cited**

U.S. PATENT DOCUMENTS

4,370,711	1/1983	Smith	395/375
4,456,955	6/1984	Yanagita et al.	395/375
4,477,872	10/1984	Losq et al.	395/375
4,760,520	7/1988	Shintani et al.	395/375
4,777,592	10/1988	Jones et al.	395/375
4,814,976	3/1989	Hansen et al.	395/375
4,860,197	8/1989	Langendorf et al.	395/375
4,984,154	1/1991	Hanatani et al.	395/375
5,146,570	9/1992	Hester et al.	395/375
5,268,213	11/1993	Weiser et al.	395/375
5,283,873	2/1994	Steely, Jr. et al.	395/375
5,287,467	2/1994	Blaner et al.	395/375
5,297,281	3/1994	Emma et al.	395/650

OTHER PUBLICATIONS

"Opcode Remap & Compression in Hard–Wired RISC Microprocessor," IBM Technical Disclosure Bulletin, vol. 32 No. 10A, Mar. 1990, N.Y., p. 349.

Lilja; "Reducing The Branch Penalty in Pipelined Processors". IEEE Jul. 1988, pp. 47–55.

Dwyer et al., "A Fast Instruction Dispatch Unit for Multiple and Out–of–Sequence Issuances" EE–CEG–87∝15, pp. 1–10 & FIGS. 1–9.

McFarling et al.; "Reducing the Cost of Branched" IEEE 1986, pp. 396–403.

Sohi et al.; "Instruction Issue Logic for High Performance Interruptable Pipelined Processors"; ACM 1987; pp. 27–34.

"Branch Prediction Strategies and Branch Target Buffer Design," Computer, vol. 17, No. 2, Jan. 1984, pp. 6–21.

Primary Examiner—Krisna Lim

Attorney, Agent, or Firm—Joseph C. Redmond; Mark Wurm

[57] **ABSTRACT**

A data processing system includes branch prediction apparatus for storing branch data in a branch prediction RAM after each branch has occurred. The RAM interfaces with branch logic means which tracks whether a branch is in progress and if a branch was guessed. An operational code compression means forms each instruction into a new operation code of lesser bits and embeds a guess bit into the new operational code. Control means decode the compressed operational code as an input to an instruction execution unit whereby conditional branch occurs based on the guess bit provided a branch instruction is not in progress in the system.

9 Claims, 3 Drawing Sheets

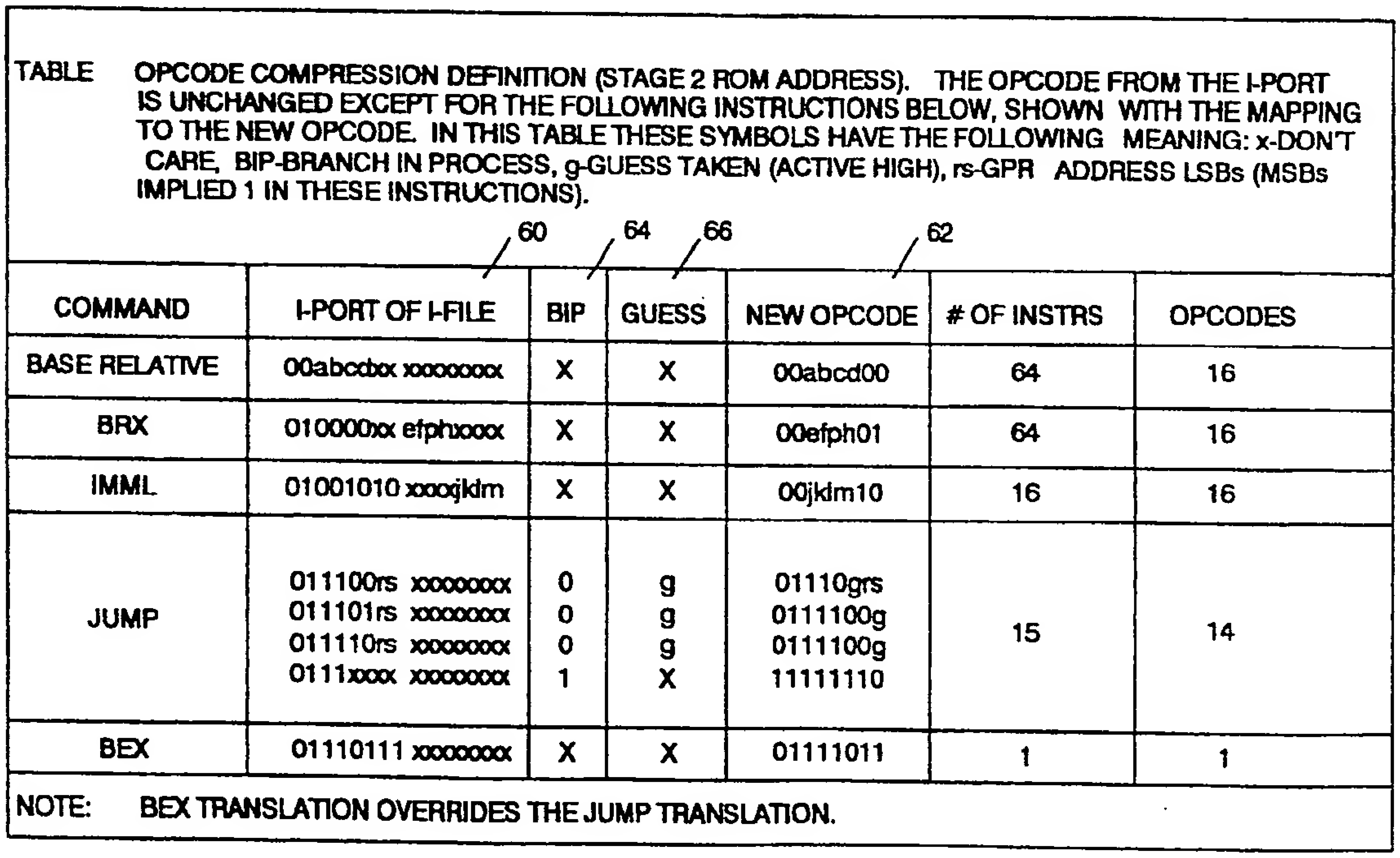


TABLE OPCODE COMPRESSION DEFINITION (STAGE 2 ROM ADDRESS). THE OPCODE FROM THE I-PORT IS UNCHANGED EXCEPT FOR THE FOLLOWING INSTRUCTIONS BELOW, SHOWN WITH THE MAPPING TO THE NEW OPCODE. IN THIS TABLE THESE SYMBOLS HAVE THE FOLLOWING MEANING: x-DON'T CARE, BIP-BRANCH IN PROCESS, g-GUESS TAKEN (ACTIVE HIGH), rs-GPR ADDRESS LSBs (MSBs IMPLIED 1 IN THESE INSTRUCTIONS).

COMMAND	I-PORT OF I-FILE	BIP	GUESS	NEW OPCODE	# OF INSTRS	OPCODES
BASE RELATIVE	00abcdxx xxxxxxxx	X	X	00abcd00	64	16
BRX	010000xx efphxxxx	X	X	00efph01	64	16
IMML	01001010 xxxxjklm	X	X	00jklm10	16	16
JUMP	011100rs xxxxxxxx 011101rs xxxxxxxx 011110rs xxxxxxxx 0111xxxx xxxxxxxx	0 0 0 1	g g g X	01110grs 0111100g 0111100g 11111110	15	14
BEX	01110111 xxxxxxxx	X	X	01111011	1	1

NOTE: BEX TRANSLATION OVERRIDES THE JUMP TRANSLATION.

	Document ID	U	Title	Current OR
22	US 5933624 A	☒	Synchronized MIMD multi-processing system and method inhibiting instruction fetch at other processors while one processor services an interrupt	709/400
23	US 5922066 A	☒	Multifunction data aligner in wide data width processor	712/204
24	US 5918032 A	☒	Computer architecture capable of concurrent issuance and execution of general purpose multiple instructions	712/215
25	US 5881307 A	☒	Deferred store data read with simple anti-dependency pipeline inter-lock control in superscalar processor	712/23
26	US 5881272 A	☒	Synchronized MIMD multi-processing system and method inhibiting instruction fetch at other processors on write to program counter of one processor	709/400
27	US 5815420 A	☒	Microprocessor arithmetic logic unit using multiple number representations	708/524
28	US 5809288 A	☒	Synchronized MIMD multi-processing system and method inhibiting instruction fetch on memory access stall	709/400
29	US 5805913 A	☒	Arithmetic logic unit with conditional register source selection	712/209
30	US 5768609 A	☒	Reduced area of crossbar and method of operation	712/11
31	US 5761726 A	☒	Base address generation in a multi-processing system having plural memories with a unified address space corresponding to each processor	711/147
32	US 5758195 A	☒	Register to memory data transfers with field extraction and zero/sign extension based upon size and mode data corresponding to employed address register	712/300
33	US 5752064 A	☒	Computer architecture capable of concurrent issuance and execution of general purpose multiple instructions	712/23
34	US 5742538 A	☒	Long instruction word controlling plural independent processor operations	708/620
35	US 5734880 A	☒	Hardware branching employing loop control registers loaded according to status of sections of an arithmetic logic unit divided into a plurality of sections	712/221
36	US 5727225 A	☒	Method, apparatus and system forming the sum of data in plural equal sections of a single data word	712/224
37	US 5717946 A	☒	Data processor	712/225
38	US 5712999 A	☒	Address generator employing selective merge of two independent addresses	711/211
39	US 5696959 A	☒	Memory store from a selected one of a register pair conditional upon the state of a selected status bit	712/245
40	US 5696958 A	☒	Method and apparatus for reducing delays following the execution of a branch instruction in an instruction pipeline	712/235

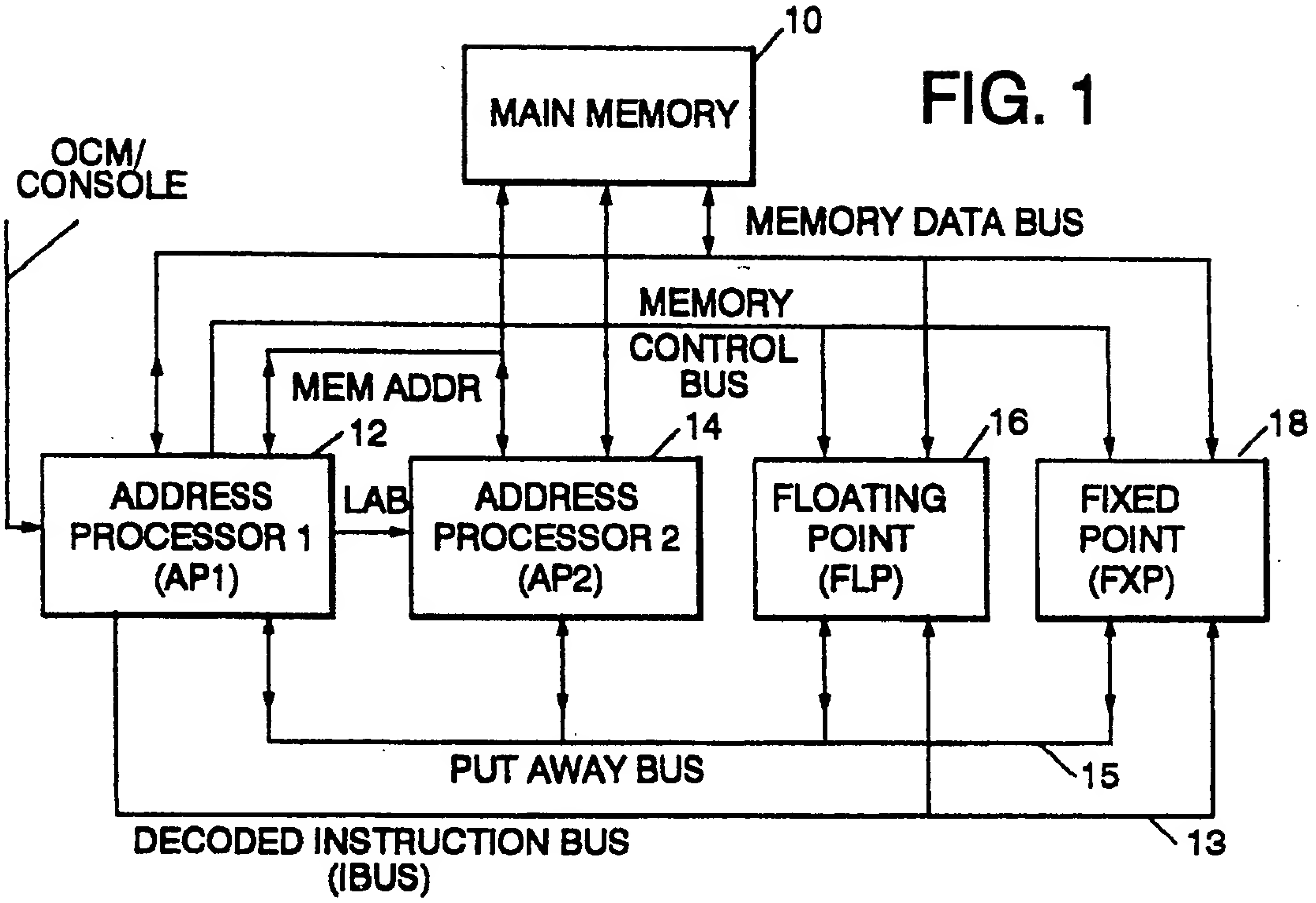
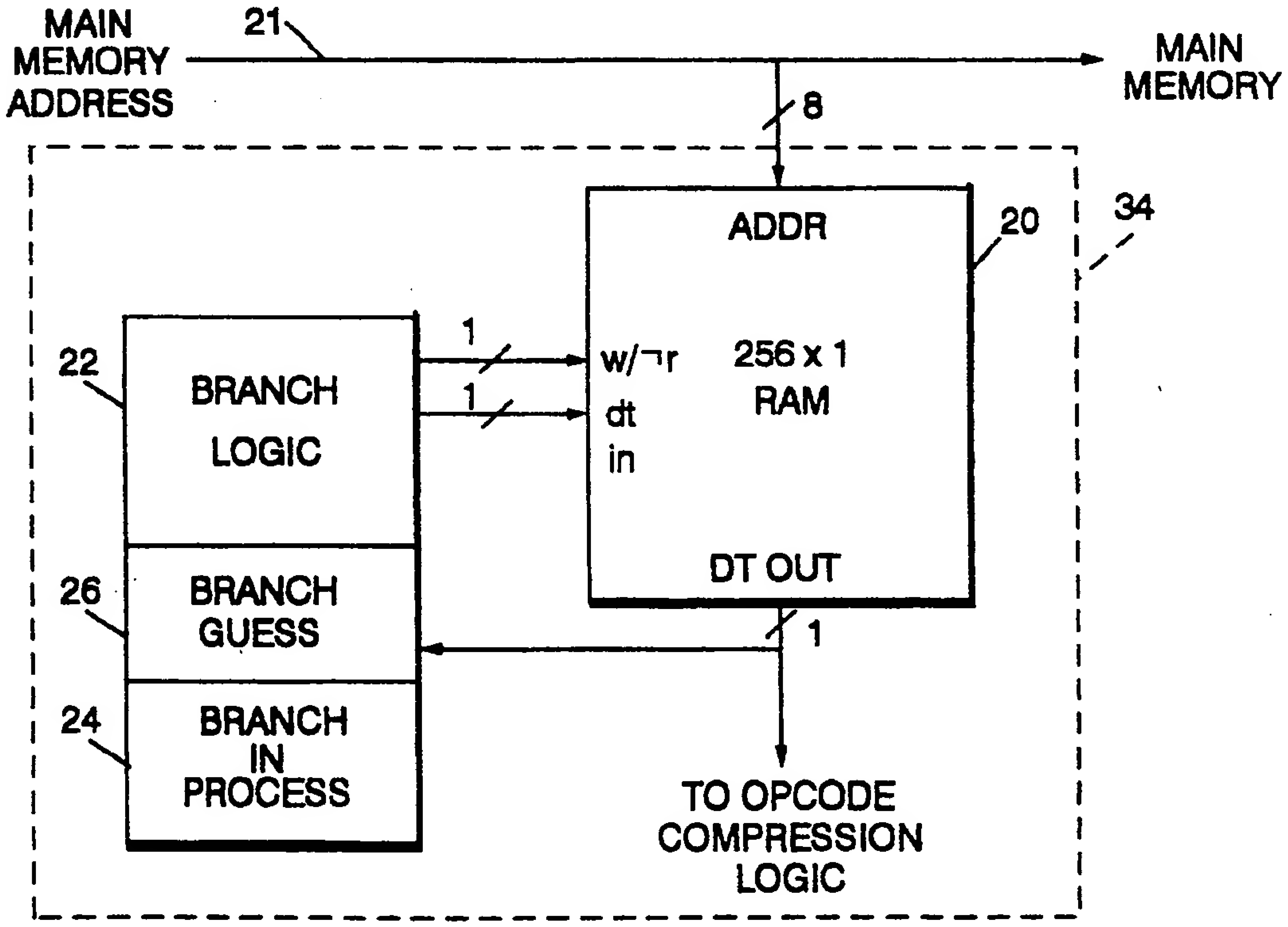

10
MAIN MEMORY
OCM/
CONSOLE
MEMORY DATA BUS
MEMORY
CONTROL
BUS
MEM ADDR
12
14
16
18
ADDRESS
PROCESSOR 1
(AP1)
LAB
ADDRESS
PROCESSOR 2
(AP2)
FLOATING
POINT
(FLP)
FIXED
POINT
(FXP)
PUT AWAY BUS
15
DECODED INSTRUCTION BUS
(IBUS)
13

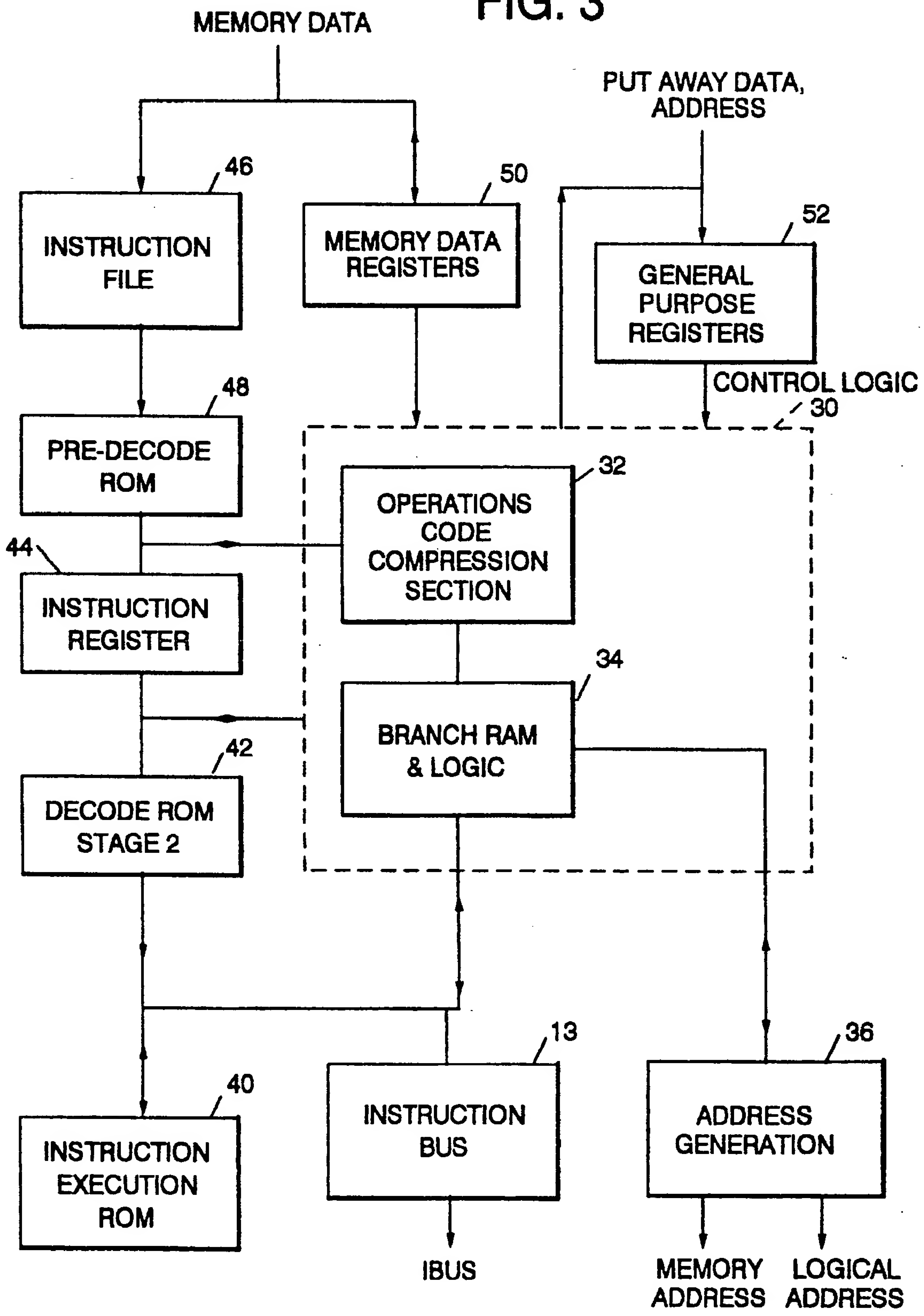
FIG. 1

FIG. 2


MAIN
MEMORY
ADDRESS
21
MAIN
MEMORY
8
ADDR
20
34
22
BRANCH
LOGIC
1
1
w/¬r
dt
in
256 x 1
RAM
DT OUT
26
BRANCH
GUESS
1
24
BRANCH
IN
PROCESS
TO OPCODE
COMPRESSION
LOGIC

	Document ID	U	Title	Current OR
41	US 5696954 A	☒	Three input arithmetic logic unit with shifting means at one input forming a sum/difference of two inputs logically anded with a third input logically ored with the sum/difference logically anded with an inverse of the third input	712/221
42	US 5696913 A	☒	Unique processor identifier in a multi-processing system having plural memories with a unified address space corresponding to each processor	710/131
43	US 5694348 A	☒	Method apparatus and system for correlation	708/525
44	US 5689695 A	☒	Conditional processor operation based upon result of two consecutive prior processor operations	712/234
45	US 5680339 A	☒	Method for rounding using redundant coded multiply result	708/493
46	US 5644699 A	☒	Memory apparatus and data processor using the same	714/7
47	US 5644524 A	☒	Iterative division apparatus, system and method employing left most one's detection and left most one's detection with exclusive or	708/655
48	US 5644522 A	☒	Method, apparatus and system for multiply rounding using redundant coded multiply result	708/551
49	US 5640578 A	☒	Arithmetic logic unit having plural independent sections and register storing resultant indicator bit from every section	712/221
50	US 5634065 A	☒	Three input arithmetic logic unit with controllable shifter and mask generator	708/230
51	US 5628024 A	☒	Computer architecture capable of concurrent issuance and execution of general purpose multiple instructions	712/23
52	US 5613146 A	☒	Reconfigurable SIMD/MIMD processor using switch matrix to allow access to a parameter memory by any of the plurality of processors	712/20
53	US 5606677 A	☒	Packed word pair multiply operation forming output including most significant bits of product and other bits of one input	712/208
54	US 5606520 A	☒	Address generator with controllable modulo power of two addressing capability	708/491
55	US 5600847 A	☒	Three input arithmetic logic unit with mask generator	712/36
56	US 5596763 A	☒	Three input arithmetic logic unit forming mixed arithmetic and boolean combinations	708/670
57	US 5596519 A	☒	Iterative division apparatus, system and method employing left most one's detection and left most one's detection with exclusive OR	708/655
58	US 5592405 A	☒	Multiple operations employing divided arithmetic logic unit and multiple flags register	708/518
59	US 5590350 A	☒	Three input arithmetic logic unit with mask generator	712/36
60	US 5574941 A	☒	Computer architecture capabie of concurrent issuance and execution of general purpose multiple instruction	712/215

FIG. 3


MEMORY DATA
PUT AWAY DATA, ADDRESS
46
INSTRUCTION FILE
50
MEMORY DATA REGISTERS
52
GENERAL PURPOSE REGISTERS
CONTROL LOGIC
30
48
PRE-DECODE ROM
32
OPERATIONS CODE COMPRESSION SECTION
44
INSTRUCTION REGISTER
34
BRANCH RAM & LOGIC
42
DECODE ROM STAGE 2
13
36
40
INSTRUCTION BUS
ADDRESS GENERATION
INSTRUCTION EXECUTION ROM
IBUS
MEMORY ADDRESS
LOGICAL ADDRESS

	Document	U	Title	Current OR
61	USD 5522083 A	☒	Reconfigurable multi-processor operating in SIMD mode with one processor fetching instructions for use by remaining processors	712/22
62	US 5512896 A	☒	Huffman encoding method, circuit and system employing most significant bit change for size detection	341/65
63	US 5509129 A	☒	Long instruction word controlling plural independent processor operations	712/203
64	US 5493524 A	☒	Three input arithmetic logic unit employing carry propagate logic	708/709
65	US 5485411 A	☒	Three input arithmetic logic unit forming the sum of a first input anded with a first boolean combination of a second input and a third input plus a second boolean combination of the second and third inputs	708/230
66	US 5479620 A	☒	Control unit modifying micro instructions for one cycle execution	712/226
67	US 5479166 A	☒	Huffman decoding method, circuit and system employing conditional subtraction for conversion of negative numbers	341/65
68	US 5471592 A	☒	Multi-processor with crossbar link of processors and memories and method of operation	709/213
69	US 5465224 A	☒	Three input arithmetic logic unit forming the sum of a first Boolean combination of first, second and third inputs plus a second Boolean combination of first, second and third inputs	708/236
70	US 5446651 A	☒	Split multiply operation	708/630
71	US 5442581 A	☒	Iterative division apparatus, system and method forming plural quotient bits per iteration	708/653
72	US 5420809 A	☒	Method of operating a data processing apparatus to compute correlation	708/200
73	US 5410649 A	☒	Imaging computer system and network	345/505
74	US 5390355 A	☒	Computer architecture capable of concurrent issuance and execution of general purpose multiple instructions	712/206
75	US 5381531 A	☒	Data processor for selective simultaneous execution of a delay slot instruction and a second subsequent instruction the pair following a conditional branch instruction	712/235
76	US 5371896 A	☒	Multi-processor having control over synchronization of processors in mind mode and method of operation	712/20
77	US 5371860 A	☒	Programmable controller	710/22
78	US 5339447 A	☒	Ones counting circuit, utilizing a matrix of interconnected half-adders, for counting the number of ones in a binary string of image data	377/82
79	US 5239654 A	☒	Dual mode SIMD/MIMD processor providing reuse of MIMD instruction memories as data memories when operating in SIMD mode	712/20
80	US 5226125 A	☒	Switch matrix having integrated crosspoint logic and method of operation	710/132

FIG. 4

TABLE OPCODE COMPRESSION DEFINITION (STAGE 2 ROM ADDRESS). THE OPCODE FROM THE I-PORT IS UNCHANGED EXCEPT FOR THE FOLLOWING INSTRUCTIONS BELOW, SHOWN WITH THE MAPPING TO THE NEW OPCODE. IN THIS TABLE THESE SYMBOLS HAVE THE FOLLOWING MEANING: x-DON'T CARE, BIP-BRANCH IN PROCESS, g-GUESS TAKEN (ACTIVE HIGH), rs-GPR ADDRESS LSBs (MSBs IMPLIED 1 IN THESE INSTRUCTIONS).

	60	64	66	62		
COMMAND	I-PORT OF I-FILE	BIP	GUESS	NEW OPCODE	# OF INSTRS	OPCODES
BASE RELATIVE	00abcdxx xxxxxxxx	X	X	00abcd00	64	16
BRX	010000xx efphxxxx	X	X	00efph01	64	16
IMML	01001010 xxxxjklm	X	X	00jklm10	16	16
JUMP	011100rs xxxxxxxx 011101rs xxxxxxxx 011110rs xxxxxxxx 0111xxxx xxxxxxxx	0 0 0 1	g g g X	01110grs 0111100g 0111100g 11111110	15	14
BEX	01110111 xxxxxxxx	X	X	01111011	1	1

NOTE: BEX TRANSLATION OVERRIDES THE JUMP TRANSLATION.

	Document ID	U	Title	Current OR
81	US 5212777 A	☒	Multi-processor reconfigurable in single instruction multiple data (SIMD) and multiple instruction multiple data (MIMD) modes and method of operation	712/229
82	US 5197140 A	☒	Sliced addressing multi-processor and method of operation	711/220
83	US 5075844 A	☒	Paired instruction processor precise exception handling mechanism	712/218
84	US 5072364 A	☒	Method and apparatus for recovering from an incorrect branch prediction in a processor that executes a family of instructions in parallel	712/215
85	US 4722050 A	☒	Method and apparatus for facilitating instruction processing of a digital computer	712/205
86	US 4415969 A	☒	Macroinstruction translator unit for use in a microprocessor	712/227
87	US 4298927 A	☒	Computer instruction prefetch circuit	712/207

DATA PROCESSING SYSTEM HAVING PREDICTION BY USING AN EMBEDDED GUESS BIT OF REMAPPED AND COMPRESSED OPCODES

This invention was made with Government support under contract number F29601-87-C-0006, awarded by the Department of the Air Force. The Government has certain rights in this invention.

BACKGROUND OF THE INVENTION

1. Technical Field

The invention disclosed broadly relates to digital computer processing systems and more particularly to pipelined data processing systems including branch prediction.

2. Background Art

Data processing systems generally include a central processor, associated storage systems and peripheral devices and interfaces. Typically the main memory consists of relatively low cost, high capacity, digital storage devices. The peripheral devices may be, for example, nonvolatile, semi-permanent storage media such as magnetic disks and magnetic tape drives. In order to carry out tasks, the central processor of such a system executes a succession of instructions which operate on the data. The succession of instructions and the data those instructions reference are referred to as a program.

In the operation of such systems, programs are initially brought to an intermediate storage area, usually in the main memory. The central processor may then interface directly to the main memory to execute the stored program. However, this procedure places limitations on performance due principally to the relative long times required in accessing that main memory. To overcome these limitations, a high speed storage system, in some cases called a cache is used to hold currently used portions of program within the central processor itself. The cache interfaces with the main memory through memory control hardware which handles program transfers between the central processor main memory and the peripheral device interfaces.

One form of computer has been developed in the prior art to concurrently process a succession of instructions in a so-called pipeline manner. In such pipeline processors, each instruction is executed in part at each of a succession of stages. After the instruction has been processed at each of the stages, the execution is complete. With this configuration, an instruction is passed from one stage to the next. That instruction is replaced by the next instruction in the program. Thus, the stages together form a pipeline which at any given time, is executing in part, a succession of instructions. Such instruction pipelines, processing a plurality of instructions in parallel, are found in several digital computing systems. These processors consist of a single pipeline of varying length and employ hardwired logic for all data manipulation. The large quantity of control logic in such machines is difficult to handle, for example, conditional branch instructions, make them extremely fast, but also very expensive.

The present invention relates to branch prediction mechanisms for handling conditional branch instructions in a computer system. When a branch instruction is encountered, it is wasteful of the computer resource to wait for resolution of the instruction before proceeding with the next programming step. Therefore, it is a known advantage to provide a prediction mechanism to predict in advance the instruction to be taken as a result of a conditional branch. If the prediction is successful, it allows a computer system to function without a delay in processing time. There is a time penalty if the prediction is incorrect. Therefore an object of the present invention is to provide an improved branch prediction mechanism with a high prediction accuracy to minimize the time loss caused by incorrect predictions.

In most pipeline processors, conditional branch instructions are resolved in the execution unit. Hence, there are several cycles of delay between the decoding of a conditional branch instruction and its execution. In an attempt to overcome the potential loss of these cycles, the decoder guesses as to which instructions to decode next. Many pipeline processors classify branches according to an instruction field. When a branch is decoded, the outcome of the branch is predicted, based on its class.

An example of a prior art branch prediction scheme is disclosed in U.S. Pat. No. 4,477,872 to Losq, et al. which patent is assigned to the assignee of the present invention. The method disclosed predicts the outcome of a conditional branch instruction based on the previous performance of the branch, rather than on the instruction fields. The prediction of the outcome of a conditional branch is performed utilizing a table which records a history of the outcome of the branch at a given memory location. The disclosed method predicts only the branch outcomes and not the address targets for prefetching an instruction. The present invention is related to patent application Ser. No. 07/783,060 entitled "Synchronizing a Prediction RAM," assigned to the assignee of the present invention, filed Oct. 25, 1991, its teachings are herein incorporated by reference. Disclosed is a high speed, pipelined CPU which breaks large execution flows into stages to allow a dramatic improvement in the system latency between registers. The multitude of stages allow better observability for testing and debugging of the overall system.

The performance enhancement of the pipeline processor is dependent on the degree to which each stage of the pipeline is kept busy processing its instructions and passing the results onto the next stage. In an ideal environment, each instruction would pass through a new stage every clock cycle. With this assumption, instruction execution time would be equal to the clock cycle time after the start-up latency has filled the pipeline. A serious degradation of pipeline performance improvement can result when branch instructions cause the pipeline to be flushed and restarted with a new instruction stream. It is desirable to know the result of a conditional branch instruction when instructions are being fetched. Unfortunately, this is not always possible, because conditional branches are often dependent on the instruction immediately preceding them in the pipeline.

OBJECTS OF THE INVENTION

It is therefore an object to provide a highly accurate branch prediction.

It is another object of the invention to provide for instruction operation compression within the computer processing unit.

SUMMARY OF THE INVENTION

The present invention employs the least significant eight bits from the memory address used to address a RAM. Assuming repeatability in programming, a decision has been made to guess that the branch will resolve in the same way the previous branch to a given address was decided. This is done by using the memory address to read a RAM which

Printed by EAST

UserID: KKim1

Computer: WS06473

Date: 11/14/2000

Time: 16:00

was written with branch data after the branch has been resolved. Rather than the entire memory address, only the lower eight bits are used. This provides a good trade-off between hardware, which dramatically increases the number of bits used to address the prediction RAM and performance of the device.

Along with branch prediction, an operations instruction code has been compressed from a 12-bit to an eight-bit mapping to provide a 160 operations to be derived from 62 operational codes. This reduces the needed ROM space from 512-byte ROM to a 256-byte ROM, which represents significant savings in hardware size and speed.

BRIEF DESCRIPTION OF THE DRAWINGS

These and other objects, features and advantages of the present invention will be more fully appreciated with reference to the accompanying figures.

FIG. 1 is a schematic diagram of a typical computer system employing central processing units tied to communication buses.

FIG. 2 is a logic diagram showing the implementation of the present invention.

FIG. 3 is a block diagram of the branch prediction RAM logic.

FIG. 4 is a table showing the operations code compression scheme of the present.

DISCUSSION OF THE PREFERRED EMBODIMENT

An example of a typical computer system embodying the present invention is shown in FIG. 1. Address processor 12 reads instructions from the main memory 10 and dispatches commands to execution elements such as fixed point processor 18 and floating point processor 16, or the address translator 14. The address processor 12 sources the instruction bus (I-bus) 13 which issues service requests to the execution elements. Any general purpose petition updating is done across the put-away bus 15.

Assuming repeatability in programming, it was decided to implement the best guess that the conditional branch would be resolved the same way that the previous conditional branch to a given address was decided. This is done by using the memory address to read a RAM that is written with branch data after the branch has been resolved. Rather than the entire memory address, only the lower eight bits are used. This provides a good trade-off between hardware, which increases dramatically with the number of bits used to address the prediction RAM, and performance.

Shown in FIG. 2 is an implementation in detail of the main memory bus 21 from which the least significant eight bits have been input into a prediction controller 20, which is a 256-bit RAM. Controller 20 interfaces with the branch logic 22. A determination of branch in progress (BIP) is made in section 24. If a branch is in progress a guess prediction is made in unit 26. The least significant 8 bits from the memory address are used to address the RAM 20. Branch logic tracks whether a branch was guessed and if a branch is currently in progress. A significant speed and hardware enhancement to the implementation of this branch prediction is the inclusion of the guess in the formation of the operations code.

Shown in FIG. 3 is a block diagram of the address processor of the present invention. Control logic 30 contains an operations code compression section 32 and a branch RAM logic 34. Address generators 36 output and receive memory and logical addresses to the computer system. Instruction bus 13 is connected to the branch RAM and logic unit 34. Instruction execution ROM 40 interfaces with the instruction bus and decodes the instructions in decode ROM 42. Instruction register 44 receives as an input memory data through precode RAM 48 from instruction file 46. The memory data in register 50 interfaces with the memory data in the logic control chip 30. Put-away bus 15 handles data and addresses at general purpose register 52 shown interfacing with the control logic 30.

The microcode for a given instruction is executed by first passing the instruction code through a pre-decode RAM 48 which produces the first microword for all instructions. Further microwords for given instructions are produced in the instruction microcode ROM 42. The use of microcodes is a characteristic of a Complex Instruction Set Computer (CISC) architecture. It allows a variety of instructions to be decoded with a minimal amount of hardware. While not as fast as hardwired solutions, the microcode ROMs have a relatively quick decode time. Imbedding the guess bit of branch prediction in the microcode address (the compressed operation code) for jump operations to be decoded leads to a fast/simple decode, including the target address consistent with the guess.

The operational code 60 is manipulated for the instructions in the opcode compression unit 32. This compressed opcode allows the guess bits to be imbedded into the opcode (decode address) without requiring a larger ROM. The decode ROM allows quick target address generation and thus, execution within the cycle time. The resulting opcode compression 62 and branch instructions 64 are shown in the table of FIG. 4. The 12-bit opcodes for the extended instructions are reduced to eight bits before entering the I register 44 which addresses the decode ROM 42. It is to be noted that for the instructions shown, 160 operations are compressed to 62 operation codes. This technique, along with the compression of input/output operations, allows 384 required instructions to be decoded from a 256-byte ROM. Avoiding the use of a 512-byte ROM, which would have been needed without compression. This represents a significant saving in hardware size and speed.

It can be seen that the guess bit 66 is only relevant to conditional branches and that a guess would only effect the operation code of a conditional branch if the CPU is not processing a previous branch, as indicated by the branch in progress unit 24. The branch logic 22 combines operation code, a prediction signal and a signal which indicates another branch is in progress.

The branch prediction algorithm disclosed has achieved an accuracy of approximately 85 percent of the instruction sets tested. This is led to an overall performance improvement of approximately seven percent. The additional hardware is easily justified by this performance improvement. The hardware was limited to 256-bit RAM, guess logic and operations code compression logic. The guess logic and the compressed opcode, are done in microcode. This allows the task to be handled with good performance in a minimum of space.

Although a specific embodiment of the present invention has been disclosed, it will be understood by those of skill in the art that the foregoing and changes in form and detail can be made therein without departing from the spirit and scope of the invention.

What is claimed is:

1. A computing machine including a main memory and

	L #	Hits	Search Text
1	L1	1244	(abbreviat$3 compress$3 compact$3) near5 (instruction opcod
2	L8	821	(base near20 (offset displacement)) near20 table
3	L9	933	(base near20 (offset displacement)) near20 memory
4	L10	87776	(traslat$3 conver$4 map$4 expan$4) near10 (table memory)
5	L11	770	(8 9) and 10
6	L12	82	(8 9) near99 10
7	L13	82	(8 9) near50 10
8	L14	141	(traslat$3 conver$4 map$4 expan$4) near50 (8 9)
9	L15	30840	(traslat$3 conver$4 map$4 expan$4) adj3 (table memory)
10	L16	43	(base near20 (offset displacement)) near20 15
11	L17	1797	(base near20 (offset displacement)) near20 (index$3 select$3 e
12	L18	8	17 near20 15
13	L19	28043	(traslat$3 conver$4 map$4 expan$4) near10 code
14	L20	3	19 near50 (8 9)
15	L21	14	(base near20 (offset displacement)) near20 19
16	L4	68	(pipelin$3 stage cycle) near50 1
17	L7	2	dual near10 1
18	L6	133	(first second) near10 1
19	L5	57	(two several couple) near10 1
20	L2	193	(expand$3 decompress$3 convert$3) near50 1
21	L3	29	(recover$3 recreat$3 transform$3 translat$3) near50 1
22	L22	444	((two dual) adj2 instruction) near10 fetch$3
23	L23	6874	instruction adj2 register
24	L24	24	22 near50 23
25	L25	87	22 near20 register not 24

employing conditional branch prediction comprising:

a prediction random access memory coupled to the main memory for receiving a selected number of least significant bits of a previously written memory address;

branch prediction logic for determining if a branch guess is in progress; generating a guess bit, if a branch is in progress, and producing a prediction based on the previously written memory address;

operational code compression means for re-mapping all processor execution instruction files into compressed operational codes and embedding the guess bit into the compressed operational codes of lesser size than original operational codes included in the instruction files; and

control logic means for interfacing between the main memory and an instruction execution read-only-memory to fetch execution instruction files from main memory, decode the instruction based on the compressed operational code including the embedded guess bit, and predict a conditional branch based on the previous written memory address and the guess bit wherein the read-only-memory is reduced in size due to the compressed operational codes including the embedded guess bit and the computing machine is improved in performance.

2. The computing machine of claim **1** wherein the branch prediction RAM is written with branch data after each time a branch has been resolved.

3. The computing machine of claim **2** wherein the operational code compression means is coupled between the branch prediction RAM and an instruction register.

4. In a data processing system, a memory, a processor, an instruction execution unit and a branch prediction mechanism for handling conditional branch instructions comprising:

a) a branch prediction RAM coupled to the memory, the branch prediction RAM receiving a selected number of least significant digits of a previously written address from the memory as an address in the RAM;

b) branch logic coupled to the branch prediction RAM for (a) determining if a branch instruction is in progress in the instruction unit, and (b) providing a guess bit if a branch is in progress;

c) an operational code compression means coupled to the memory for mapping each operation code to form a new operation code of a lesser number of bits and embedding the guess bit in each new operation code; and

d) control means interfacing between the memory and the instruction execution unit for decoding the compressed operational code and predicting a conditional branch based upon the guess bit provided a branch instruction is not in progress.

5. The data processing system of claim **3** further including a precode RAM coupled to the processor instruction files; the precode RAM generating a microcode word as an input to an instruction register, and a decode ROM coupled to the instruction register for generating further microcode words related to the microcode word, the further microcode words provided as successive inputs to the instruction execution unit.

6. The system of claim **5** wherein each operation instruction code in the system is a 12-bit word and each compressed operation code is an 8-bit word whereby the number of system instruction operations are reduced from 160 to 62 through the use of a decode ROM of lesser size than a predecode RAM.

7. A method of branch prediction in a data processing system including a memory, an instruction execution unit, and control means interfacing the memory and the instruction execution unit comprising the steps of:

a. generating a branch prediction signal in a branch prediction RAM using a selected number of least significant digits of a previously written address from the memory as an address for the RAM;

b. determining if a branch is in progress in the instruction execution unit and generating a guess bit if a branch is in progress in a branch prediction logic means;

c. combining the branch prediction signal and guess bit in the branch prediction logic;

d. compressing all processor instruction files in an operation code compression means to form new operation codes of a lesser number of bits and embedding the guess bit in the new operational codes; and

e. decoding the new operational code and predicting a conditional branch as an input to the instruction execution unit based on the guess bit.

8. The method of claim **7** further comprising the step of forming each processor execution file as a microcode word and embedding the guess bit into the word for execution provided a branch is not in progress in the system.

9. The method of claim **8** further including the step of decoding the microcode word and guess bit in a decode instruction ROM within a cycle time of the system.

* * * * *

	Document	U	Title	Current OR
1	USD 6088808 A	□	Low power consumption semiconductor integrated circuit device and microprocessor	713/324
2	US 5848268 A	☒	Data processor with branch target address generating unit	712/233
3	US 5835746 A	☒	Method and apparatus for fetching and issuing dual-word or multiple instructions in a data processing system	712/215
4	US 5832258 A	☒	Digital signal processor and associated method for conditional data operation with no condition code update	712/226
5	US 5829049 A	☒	Simultaneous execution of two memory reference instructions with only one address calculation	711/168
6	US 5799163 A	☒	Opportunistic operand forwarding to minimize register file read ports	712/205
7	US 5752014 A	☒	Automatic selection of branch prediction methodology for subsequent branch instruction based on outcome of previous branch prediction	712/240
8	US 5734913 A	☒	Low power consumption semiconductor integrated circuit device and microprocessor	713/322
9	US 5713012 A	☒	Microprocessor	712/233
10	US 5649145 A	☒	Data processor processing a jump instruction	711/213
11	US 5638524 A	☒	Digital signal processor and method for executing DSP and RISC class instructions defining identical data processing or data transfer operations	712/221
12	US 5634118 A	☒	Splitting a floating-point stack-exchange instruction for merging into surrounding instructions by operand translation	712/226
13	US 5617550 A	☒	Data processor generating jump target address of a jump instruction in parallel with decoding of the instruction	712/207
14	US 5592637 A	☒	Data processor processing a jump instruction	712/237
15	US 5590296 A	☒	Data processor processing a jump instruction	712/229
16	US 5537561 A	☒	Processor	712/23
17	US 5485587 A	☒	Data processor calculating branch target address of a branch instruction in parallel with decoding of the instruction	712/234
18	US 5457790 A	☒	Low power consumption semiconductor integrated circuit device and microprocessor	711/167
19	US 5398321 A	☒	Microcode generation for a scalable compound instruction set machine	712/216
20	US 4858105 A	☒	Pipelined data processor capable of decoding and executing plural instructions in parallel	712/235
21	US 4439827 A	☒	Dual fetch microsequencer	712/235

operation, the compressed operation length being chosen from a plurality of finite lengths, which finite lengths include at least two non-zero lengths, which of the finite lengths is chosen being dependent upon at least one feature of the operation.

2. The medium of claim 1 wherein the compressed operation length can only be one of 26, 34 and 42 for any non-null operation.

3. The medium of claim 1 wherein the at least one feature is at least one of the following:

abbreviated op code;
guarded or unguarded;
resultless;
immediate parameter with fixed number of bits; and
zeroary, unary, or binary.

4. The medium of claim 3 wherein combined operation types are aliased according to the following table

FORMAT	ALIASED TO
zeroary	unary
unary_resultless	unary
binary_resultless_short	binary_resultless
zeroary_param32_short	zeroary_param32
zeroary_param32_resultless_short	zeroary_param32_resultless
zeroary_short	unary
unary_resultless_short	unary
binary_resultless_unguarded	binary_resultless
unary_unguarded	unary
binary_param7_resultless_unguarded	binary_param7_resultless
unary_unguarded	unary
binary_param7_resultless_unguarded	binary_param7_resultless
zeroary_unguarded	unary
unary_resultless_unguarded_short	binary_unguarded_short
unary_unguarded_short	unary_short
zeroary_param32_unguarded_short	zeroary_param32
zeroary_parame32_resultless_unguarded_short	zeroary_param32_resultless
zeroary_unguarded_short	unary
unary_resultless_unguarded_short	unary
unary_long	binary
binary_long	binary
binary_resultless_long	binary
unary_param7_long	unary_param7
binary_param7_resultless_long	binary_param7_resultless
zeroary_param32_long	zeroary_param32
zeroary_param32_resultless_long	zeroary_param32_resultless
zeroary_long	binary
unary_resultless_long	binary

5. The medium of claim 3, wherein the fixed number is one of 7 and 32.

6. The medium of claim 1 comprising a plurality of such instructions, of which one instruction is a branch target, which one instruction is not compressed.

7. The medium of claim 1 wherein each operation field within each instruction includes a sub-field specifying at least one of the following: a register file address of a first operand; a register file address of a second operand; a register file address of guard information; a register file address of a result; an immediate parameter; and an op code.

8. The medium of claim 1 comprising a plurality of such instructions, each instruction comprising a format field for specifying a plurality of respective formats, one respective format for each operation of a succeeding instruction.

9. The medium of claim 8, wherein the compressed format comprises a format field specifying issue slots of the VLIW processor to be used by some instruction.

10. The medium of claim 9 comprising at least one field specifying the operation.

11. The medium of 10 wherein the at least one field specifying the operation comprises at least one byte aligned sub-field.

12. The medium of claim 10 further comprising at least one operation part sub-field located in a same byte with the format field.

13. The medium of claim 12 wherein the format field specifies that more than a threshold quantity of issue slots are to be used and further comprising at least one first operation part sub-field located in a same byte with the format field, a plurality of sub-fields specifying operations, and at least one second operation part sub-field located in a byte separate from the other sub-fields.

14. The medium of 9 wherein the format field has 2*N bits, where N is the number of issue slots.

15. The medium of claim 9 wherein the instruction takes up no more than 32 bytes.

16. The medium of claim 9 formatted as follows

```
<instruction> ::=
    <instruction start>
    <instruction middle>
    <instruction end>
    <instruction extension>
<instruction start> ::=
    <Format:2*N>{<padding:1>}V2{<2-bit operation part:2>}V1{<24-
    bit operation part :24>}V1
<instruction middle> ::= {{<2-bit operation part:2>}4 {24-bit
    operation part:24>}4}V3
<instruction end> ::= {<padding:1>}V5{<2-bit operation
part:2>}V4    {24-bit operation part:24>}V4
<instruction extension>::={<operationextension:0/8/16>}S
<padding>::= "0"
```

Wherein the variables used above are defined as follows:

N=the number of issue slots of the machine, N>0
S=the number of issue slots used in this instruction (0≦S≦N)
C1=4–(N mod 4)
If (S≦C1) then V1=S and V2=2*(C1−V1)
If (S>C1) then V1=C1 and V2=0
V3=(S−V1) div 4
V4=(S−V1) mod 4
If (V4>0) then V5=2*(4−V4) else V5=0

Explanation of notation

::=	means	"is defined as"
<field name:number>	means	the field indicated before the colon has the number of bits indicated after the colon.
{<field name>}number	means	the field indicated in the angle brackets and braces is repeated the number of times indicated after the braces
"0"	means	the character "0"
"div"	means	integer divide
"mod"	means	modulo
:0/8/16	means	that the field is 0, 8, or 16 bits long.

17. The medium of claim 9 containing an operation which is encoded in 26, 34 or 42 bits, wherein

if the operation is 26 bits, it is one of
binary unguarded short;
unary immediate 7-bit parameter unguarded operation;
binary unguarded immediate 7-bit operand resultless short; and

	Document	U	Title	Current OR
22	USD 4346437 A	☒	Microcomputer using a double opcode instruction	712/205
23	US 3626427 A	☒	LARGE-SCALE DATA PROCESSING SYSTEM	712/244
24	US 3591786 A	☒	PREDICTED ITERATION IN DECIMAL DIVISION	708/652

unary short;

if the operation is 34 bits, it is one of
binary short;
unary immediate 7-bit parameter resultless short;
binary unguarded;
unary immediate 7-bit parameter unguarded; and
unary; and

if the operation is encoded in 42 bits, it is one of

binary immediate 7-bit parameter resultless;
binary;
unary immediate 7-bit parameter;
zeroary immediate 32-bit parameter; and
zeroary, immediate 32-bit parameter resultless.

18. The medium of claim **9** wherein the operations are encoded according to the following table:

	bit position						
	24-bit operation part				2-bit part	Extension	
name	0–6	7–13	14–20	21–23	24–25	26–34–41	Size
26-format:							
<binary-unguarded-short>	src1[0:6]	src2[0:6]	dst[0:6]	opcode[0:2]	opcode[3:4]		26
<unary-param7-unguarded-short>	src1[0:6]	param[0:6]	dst[0:6]	opcode[0:2]	opcode[3:4]		26
<binary-unguarded-param7-resultless-short>	src1[0:6]	src2[0:6]	param[0:6]	opcode[0:2]	opcode[3:4]		26
<unary-short>	src1[0:6]	dst[0:6]	guard[0:6]	opcode[0:2]	opcode[3:4]		26
34-format:							
<binary-short>	src1[0:6]	src2[0:6]	guard[0:6]	opcode[0:2]	opcode[3:4]	dst[0:6] 0	34
<unary-param-7-short>	src1[0:6]	param[0:6]	guard[0:6]	opcode[0:2]	opcode[3:4]	dst[0:6] 0	34
<binary-param7-resultless-short>	src1[0:6]	src2[0:6]	guard[0:6]	opcode[0:2]	opcode[3:4]	param [0:6] 0	34
<binary-unguarded>	src1[0:6]	src2[0:6]	dst[0:6]	opcode[0.2]	opcode[3:4]	opcode[5:7] XL011	34
<binary-resultless>	src1[0:6]	src2[0:6]	guard[0:6]	opcode[0:2]	opcode[3:4]	opcode[5:7] X1001	34
<unary-param7-unguarded>	src1[0:6]	param[0:6]	dst[0:6]	opcode[0:2]	opcode[3:4]	opcode[5:7] SL111	34
<unary>	src1[0:6]	dst[0:6]	guard[0:6]	opcode[0:2]	opcode[3:4]	opcode[5:7] XL101	34
42-format							
<binary-param7-resultless>	src1[0:6]	src2[0:6]	guard[0:6]	opcode[0:2]	opcode[3:4]	opcode[5:7] SXX100 param[0:6]	42
<binary>	src1[0:6]	src2[0:6]	guard{[0:6]	opcode[0:2]	opcode[3:4]	opcode[5:7] XL0101 dst[0:6]	42
<unary-param7>	src1[0:6]	param[0:6]	guard[0:6]	opcode[0:2]	opcode[3:4]	opcode[5:7] SL1101 dst[0:6]	42
<zeroary-param32>	param [7:13]	param[0:6]	dst[0:6]	param [14:16]	param [17:18]	param [19:23] XX1 param [24:31]	42
<zeroary-param32-resultless>	param [7:13]	param[0:6]	guard[0:6]	param [14:16]	param [17:18]	param [19:23] 000 param [24:31]	42
<zeroary-param32-resultless>	param [7:13]	param[0:6]	guard[0:6]	param [14:16]	param [17:18]	param [19:23] 100 param [24:31]	42

	L #	Hits	Search Text
1	L1	1244	(abbreviat$3 compress$3 compact$3) near5 (instruction opcod
2	L3	29	(recover$3 recreat$3 transform$3 translat$3) near50 1
3	L8	821	(base near20 (offset displacement)) near20 table
4	L9	933	(base near20 (offset displacement)) near20 memory
5	L10	87776	(traslat$3 conver$4 map$4 expan$4) near10 (table memory)
6	L11	770	(8 9) and 10
7	L12	82	(8 9) near99 10
8	L13	82	(8 9) near50 10
9	L14	141	(traslat$3 conver$4 map$4 expan$4) near50 (8 9)
10	L15	30840	(traslat$3 conver$4 map$4 expan$4) adj3 (table memory)
11	L16	43	(base near20 (offset displacement)) near20 15
12	L17	1797	(base near20 (offset displacement)) near20 (index$3 select$3 e
13	L18	8	17 near20 15
14	L19	28043	(traslat$3 conver$4 map$4 expan$4) near10 code
15	L20	3	19 near50 (8 9)
16	L21	14	(base near20 (offset displacement)) near20 19
17	L4	68	(pipelin$3 stage cycle) near50 1
18	L7	2	dual near10 1
19	L6	133	(first second) near10 1
20	L5	57	(two several couple) near10 1

42-format:							
<binary-param7-resultless>	src1[0:6]	src2[0:6]	guard[0:6]	opcode[0:2]	opcode[3:4]	opcode[5:7]SXX100param[0:6]	42
<binary>	src1[0:6]	src2[0:6]	guard[0:6]	opcode[0:2]	opcode[3:4]	opcode[5:7]XL0101 dst[0:6]	42
<unary-param7>	src1[0:6]	param[0:6]	guard[0:6]	opcode[0:2]	opcode[3:4]	opcode[5:7]SL1101 dst[0:6]	42
<zeroary-param32>	param[7:13]	param[0:6]	dst[0:6]	param[14:16]	param[17:18]	param[19:23]XX1 param[24:31]	42
<zeroary-param32-resultless>	param[7:13]	param[0:6]	guard[0:6]	param[14:16]	param[17:18]	param[19:23]000 param[24:31]	42
<zeroary-param32-resultless>	param[7:13]	param[0:6]	guard[0:6]	param[14:16]	param[17:18]	param[19:23]100 param[24:31]	42

Note:
S: signed/unsigned format bit for parametric operations; S=1 if signed, S=0 if unsigned
L: latency format bit; L=0 if (latency=1 and this is not a resultless operation) else L=1
X: undefined value

FIG. 5b

	Document	U	Title	Current OR
1	US 6128755 A	☐	Fault-tolerant multiple processor system with signature voting	714/715
2	US 6128094 A	☒	Printer having processor with instruction cache and compressed program store	358/1.15
3	US 6101592 A	☐	Methods and apparatus for scalable instruction set architecture with dynamic compact instructions	712/20
4	US 6096089 A	☒	Power simulation system, power simulation method and computer-readable recording medium for recording power simulation program	703/18
5	US 6044450 A	☒	Processor for VLIW instruction	712/24
6	US 5983335 A	☒	Computer system having organization for multiple condition code setting and for testing instruction out-of-order	712/23
7	US 5960465 A	☒	Apparatus and method for directly accessing compressed data utilizing a compressed memory address translation unit and compression descriptor table	711/208
8	US 5938759 A	☒	Processor instruction control mechanism capable of decoding register instructions and immediate instructions with simple configuration	712/209
9	US 5905893 A	☒	Microprocessor adapted for executing both a non-compressed fixed length instruction set and a compressed variable length instruction set	717/5
10	US 5896519 A	☒	Apparatus for detecting instructions from a variable-length compressed instruction set having extended and non-extended instructions	712/213
11	US 5892847 A	☒	Method and apparatus for compressing images	382/232
12	US 5881308 A	☒	Computer organization for multiple and out-of-order execution of condition code testing and setting instructions out-of-order	712/23
13	US 5881260 A	☒	Method and apparatus for sequencing and decoding variable length instructions with an instruction boundary marker within each instruction	712/210
14	US 5867681 A	☒	Microprocessor having register dependent immediate decompression	712/208
15	US 5860152 A	☒	Method and apparatus for rapid computation of target addresses for relative control transfer instructions	711/213
16	US 5822578 A	☒	System for inserting instructions into processor instruction stream in order to perform interrupt processing	712/244
17	US 5819058 A	☒	Instruction compression and decompression system and method for a processor	712/210
18	US 5794010 A	☒	Method and apparatus for allowing execution of both compressed instructions and decompressed instructions in a microprocessor	703/20
19	US 5784585 A	☒	Computer system for executing instruction stream containing mixed compressed and uncompressed instructions by automatically detecting and expanding compressed instructions	712/209
20	US 5754746 A	☒	Multi-bit per pixel compression/decompression using parallel encoded streams	358/1.15

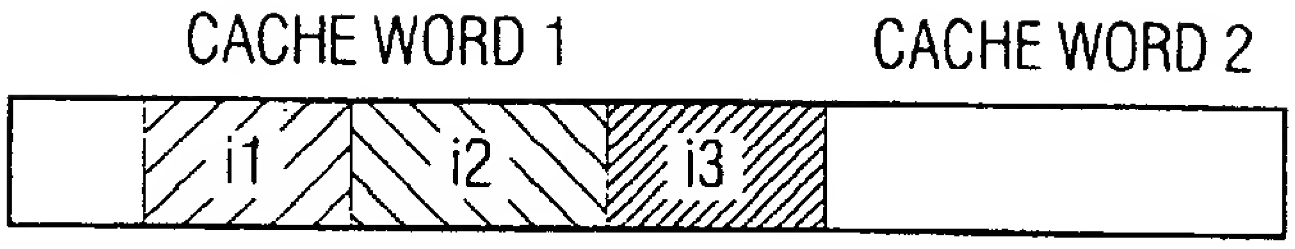


FIG. 2a

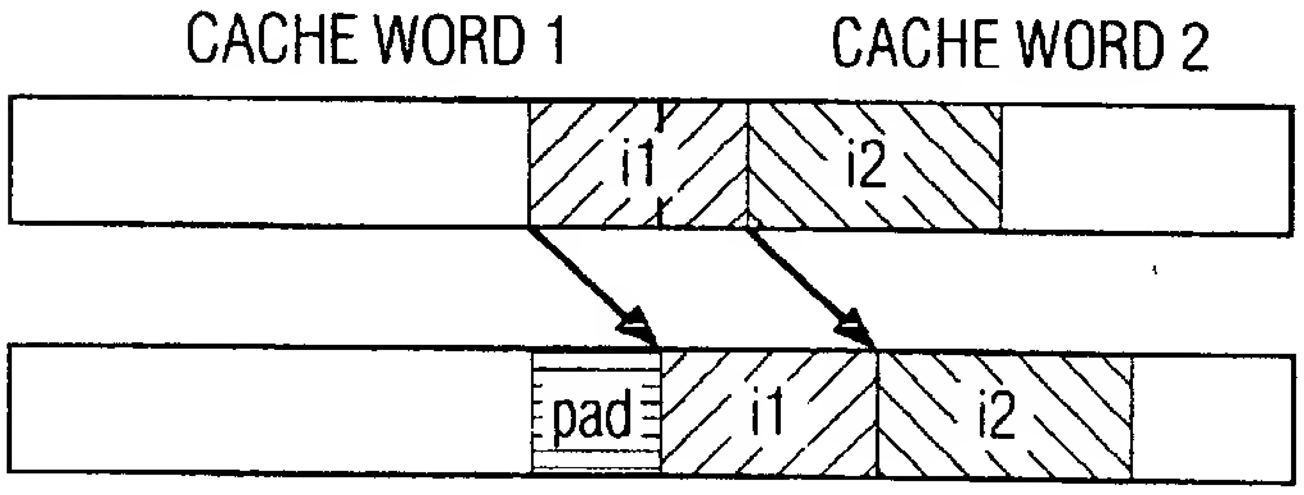


FIG. 2b

FIG. 2c

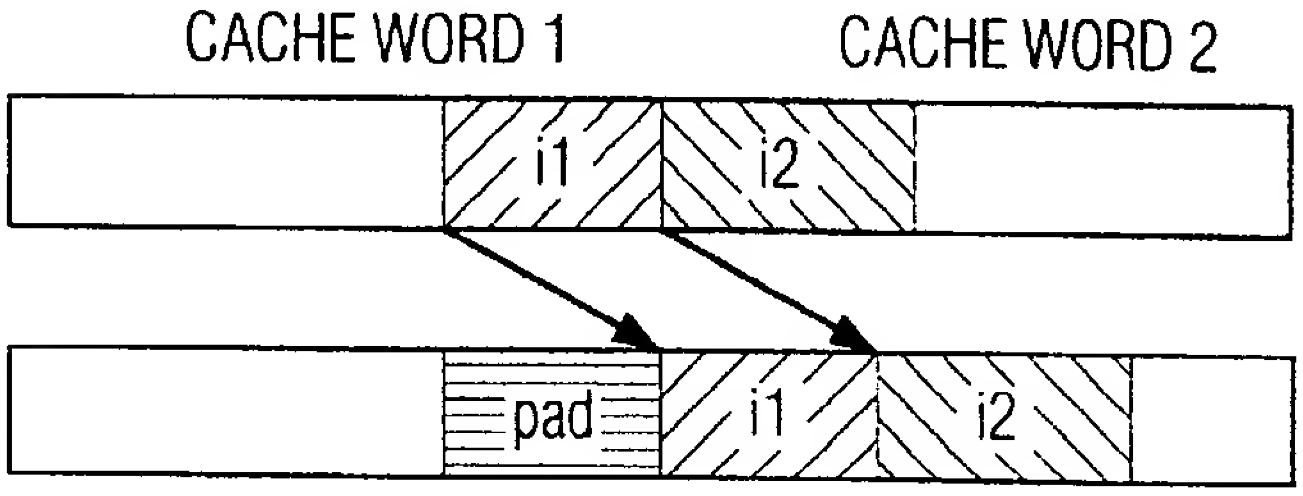


FIG. 2d

FIG. 2e

	Docu ment	U	Title	Current OR
21	USD 56690 11 A	☒	Partially decoded instruction cache	712/23
22	US 56301 57 A	☒	Computer organization for multiple and out-of-order execution of condition code testing and setting instructions	712/23
23	US 56300 85 A	☒	Microprocessor with improved instruction cycle using time-compressed fetching	712/207
24	US 56219 07 A	☒	Microprocessor with memory storing instructions for time-compressed fetching of instruction data for a second cycle within a first machine cycle	712/207
25	US 55442 47 A	☒	Transmission and reception of a first and a second main signal component	381/27
26	US 55108 57 A	☒	Motion estimation coprocessor	348/699
27	US 54817 51 A	☒	Apparatus and method for storing partially-decoded instructions in the instruction cache of a CPU having multiple execution units	712/213
28	US 54816 43 A	☒	Transmitter, receiver and record carrier for transmitting/receiving at least a first and a second signal component	704/227
29	US 54483 10 A	☒	Motion estimation coprocessor	348/699
30	US 53921 26 A	☒	Airborne thermal printer	358/296
31	US 53778 25 A	☒	Compact disc storage case	206/232
32	US 53236 18 A	☒	Heat storage type air conditioning apparatus	62/149
33	US 53234 88 A	☒	Memory access method and circuit in which access timing to a memory is divided into N periods to be accessed from N access request sources	358/1.16
34	US 52066 60 A	☒	Airborne thermal printer	347/218
35	US 52028 87 A	☒	Access control method for shared duplex direct access storage device and computer system therefor	714/8
36	US 49244 31 A	☒	Keyboard located indicia for instructing a multi-mode programmable computer having alphanumeric capabilities from a few keyboard keys	708/146
37	US 48736 30 A	☒	Scientific processor to support a host processor referencing common memory	712/3
38	US 48356 79 A	☒	Microprogram control system	712/212
39	US 48335 99 A	☒	Hierarchical priority branch handling for parallel execution in a parallel processor	712/236
40	US 48036 20 A	☒	Multi-processor system responsive to pause and pause clearing instructions for instruction execution control	712/203
41	US 47991 46 A	☒	System for displaying graphic information on video screen employing video display processor	710/260

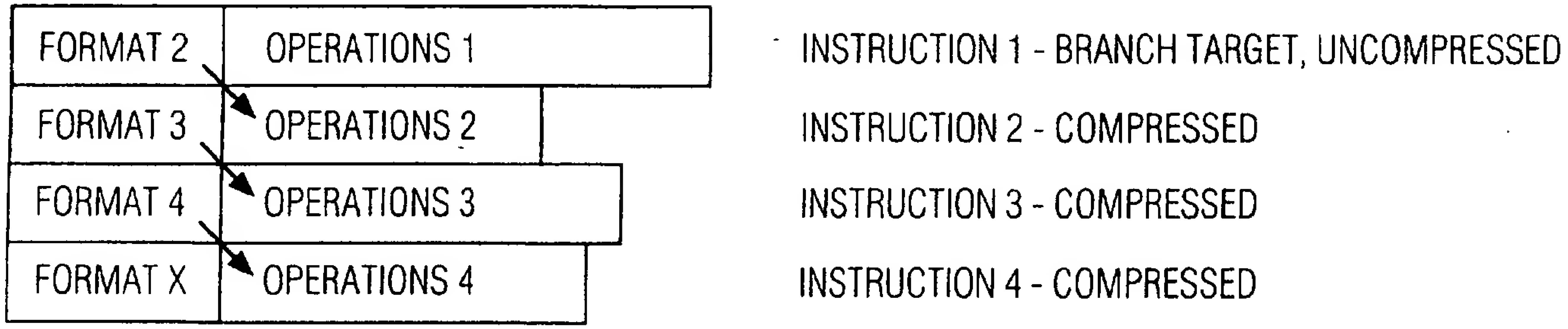

FORMAT 2
OPERATIONS 1
INSTRUCTION 1 - BRANCH TARGET, UNCOMPRESSED
FORMAT 3
OPERATIONS 2
INSTRUCTION 2 - COMPRESSED
FORMAT 4
OPERATIONS 3
INSTRUCTION 3 - COMPRESSED
FORMAT X
OPERATIONS 4
INSTRUCTION 4 - COMPRESSED

FIG. 3

	Document	U	Title	Current OR
42	USD 4770602 A	☒	Method of capacity controlling of multistage compressor and apparatus therefor	415/29
43	US 4768157 A	☒	Video image processing system	345/517
44	US 4571634 A	☒	Digital image information compression and decompression method and apparatus	358/261.3
45	US 4458110 A	☒	Storage element for speech synthesizer	704/211
46	US 4384170 A	☒	Method and apparatus for speech synthesizing	704/266
47	US 4380070 A	☒	Automatic circuit identifier	340/537
48	US 4309756 A	☒	Method of automatically evaluating source language logic condition sets and of compiling machine executable instructions directly therefrom	717/9
49	US 4288816 A	☒	Compressed image producing system	382/232
50	US 4214125 A	☒	Method and apparatus for speech synthesizing	704/268
51	US 4197578 A	☒	Microprogram controlled data processing system	712/211
52	US 4171536 A	☒	Microprocessor system	711/151
53	US 4144563 A	☒	Microprocessor system	712/42
54	US 4093982 A	☒	Microprocessor system	712/248
55	US 4058850 A	☒	Programmable controller	711/117
56	US 3936182 A	☒	Control arrangement for an electrostatographic reproduction apparatus	399/77
57	US 3656178 A	☒	DATA COMPRESSION AND DECOMPRESSION SYSTEM	341/87

FIG. 4a

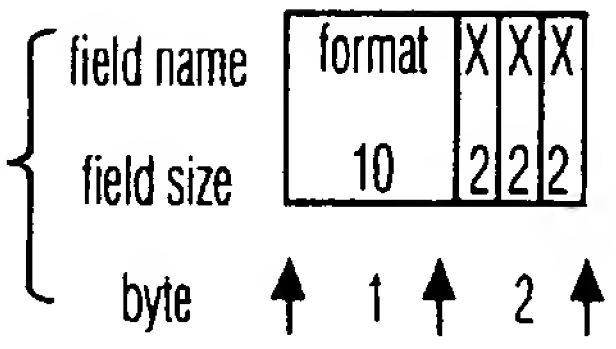


FIG. 4b

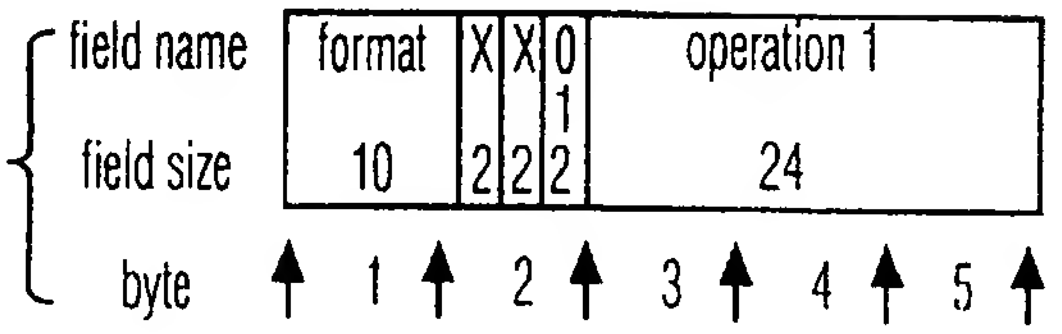


FIG. 4c

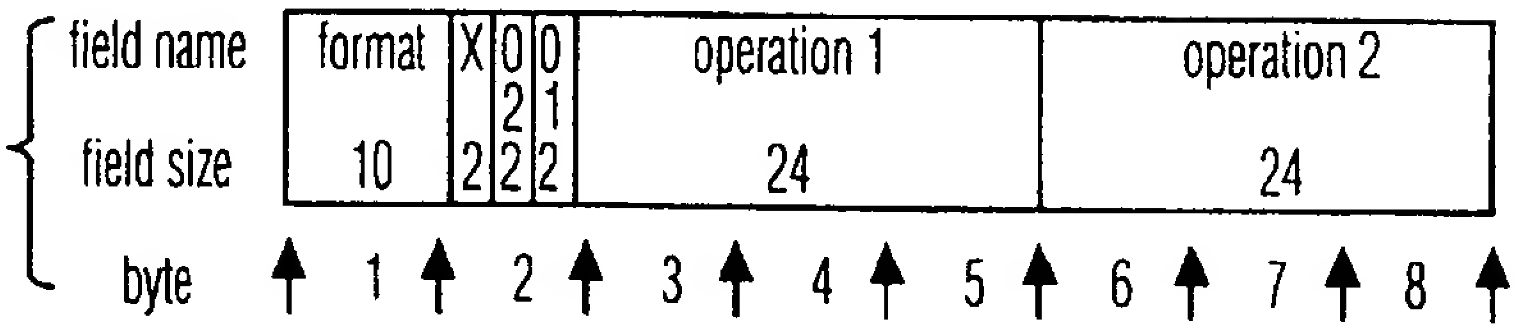


FIG. 4d

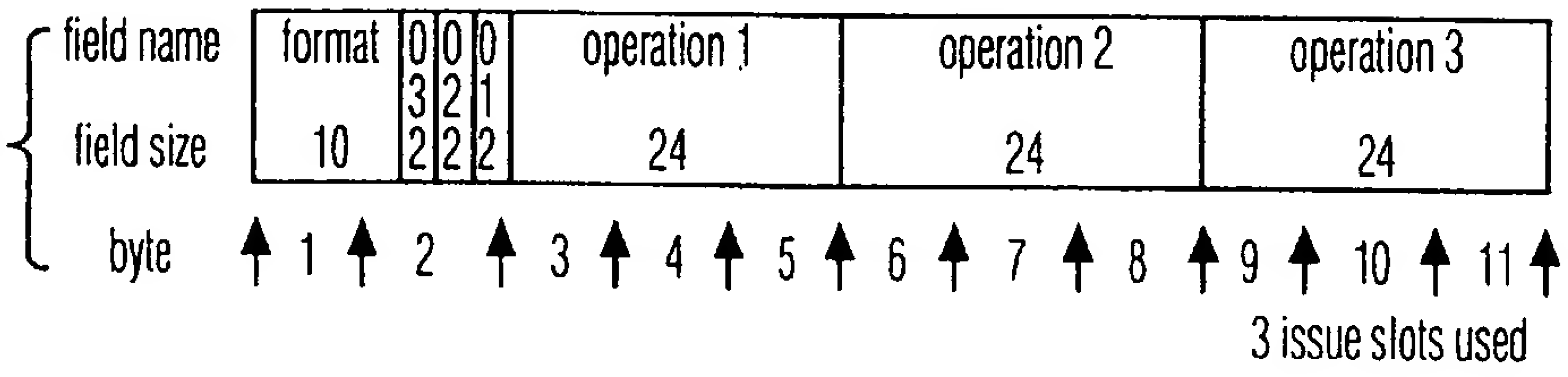


FIG. 4e

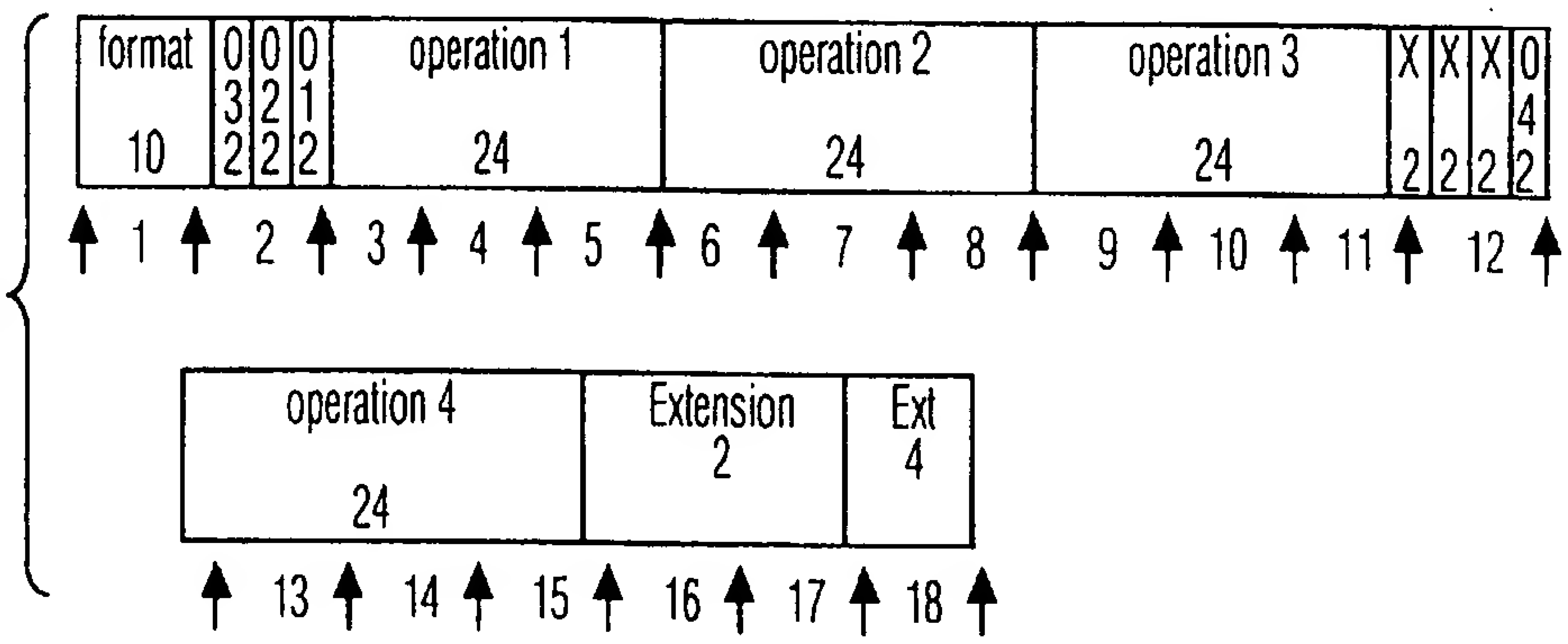
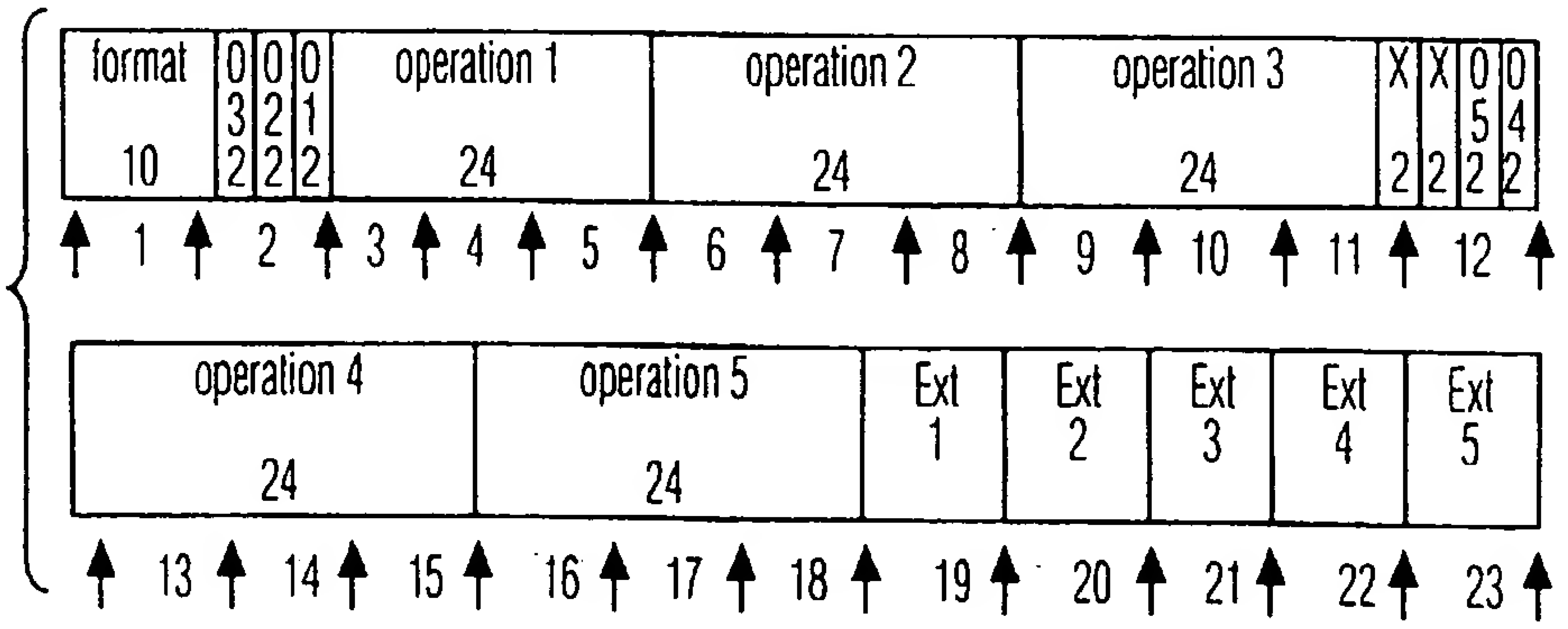


FIG. 4f



	Document	U	Title	Current OR
1	USD 6108030 A	□	Appearance inspecting device for solid formulation	348/91
2	US 6101592 A	☒	Methods and apparatus for scalable instruction set architecture with dynamic compact instructions	712/20
3	US 6044450 A	☒	Processor for VLIW instruction	712/24
4	US 5960465 A	☒	Apparatus and method for directly accessing compressed data utilizing a compressed memory address translation unit and compression descriptor table	711/208
5	US 5930508 A	☒	Method for storing and decoding instructions for a microprocessor having a plurality of function units	717/6
6	US 5907374 A	☒	Method and apparatus for processing a compressed input bitstream representing an information signal	375/240.26
7	US 5898462 A	☒	Methods of producing data storage devices for appliances which can be used to coach users in the performance of user-selected tasks	348/552
8	US 5801784 A	☒	Data storage devices	348/552
9	US 5764304 A	☒	Operation of information/entertainment centers	348/552
10	US 5694399 A	☒	Processing unit for generating signals for communication with a test access port	709/246
11	US 5646946 A	☒	Apparatus and method for selectively companding data on a slot-by-slot basis	370/442
12	US 5632024 A	☒	Microcomputer executing compressed program and generating compressed branch addresses	712/205
13	US 5610603 A	☒	Sort order preservation method used with a static compression dictionary having consecutively numbered children of a parent	341/51
14	US 5568650 A	☒	Control unit for controlling reading and writing of a magnetic tape unit	710/52
15	US 5513301 A	☒	Image compression and decompression apparatus with reduced frame memory	358/1.15
16	US 5448301 A	☒	Programmable video transformation rendering method and apparatus	348/578
17	US 5361356 A	☒	Storage isolation with subspace-group facility	711/206
18	US 5351046 A	☒	Method and system for compacting binary coded decimal data	341/62
19	US 5203352 A	☒	Polymeric foam earplug	128/864
20	US 5088031 A	☒	Virtual machine file control system which translates block numbers into virtual addresses then into real addresses for accessing main storage	709/100

the unary format is used. In that case the field for the argument is undefined.

TABLE II

OPERATION TYPE	SIZE
<binary-unguarded-short>	26
<unary-param7-unguarded-short>	26
<binary-unguarded-param7-resultless-short>	26
<unary-short>	26
<binary-short>	34
<unary-param7-short>	34
<binary-param7-resultless-short>	34
<binary-unguarded>	34
<binary-resultless>	34
<unary-param7-unguarded>	34
<unary>	34
<binary-param7-resultless>	42
<binary>	42
<unary-param7>	42
<zeroary-param32>	42
<zeroary-param32-resultless>	42

For all operations a 42-bit format is available for use in branch targets. For unary and binary-resultless operations, the <binary> format can be used. In that case, unused fields in the binary format have undefined values. Short 5-bit op codes are converted to long 8-bit op codes by padding the most significant bits with 0's. Unguarded operations get as a guard address value, the register file address of constant TRUE. For store operations the 42 bit, binary-param7-resultless> format is used instead of the regular 34 bit <binary-param7-resultless short> format (assuming store operations belong to the set of short operations).

Operation types which do not appear in table II are mapped onto those appearing in table II, according to the following table of aliases:

TABLE II

FORMAT	ALIASED TO
zeroary	unary
unary_resultless	unary
binary_resultless_short	binary_resultless
zeroary_param32_short	zeroary_param32
zeroary_param32_resultless_short	zeroary_param32_resultless
zeroary_short	unary
unary_resultless_short	unary
binary_resultless_unguarded	binary_resultless
unary_unguarded	unary
binary_param7_resultless_unguarded	binary_param7_resultless
unary_unguarded	unary
binary_param7_resultless_unguarded	binary_param7_resultless
zeroary_unguarded	unary
unary_resultless_unguarded_short	binary_unguarded_short
unary_unguarded_short	unary_short
zeroary_param32_unguarded_short	zeroary_param32
zeroary_parame32_resultless_unguarded_short	zeroary_param32_resultless
zeroary_unguarded_short	unary
unary_resultless_unguarded_short	unary
unary_long	binary
binary_long	binary
binary_resultless_long	binary
unary_param7_long	unary_param7
binary_param7_resultless_long	binary_param7_resultless
zeroary_param32_long	zeroary_param32
zeroary_param32_resultless_long	zeroary_param32_resultless
zeroary_long	binary
unary_resultless_long	binary

The following is a table of fields which appear in operations:

TABLE III

FIELD	SIZE	MEANING
src1	7	register file address of first operand
src2	7	register file address of second operand
guard	7	register file address of guard
dst	7	register file address of result
param	7/32	7 bit parameter or 32 bit immediate value
op code	5/8	5 bit short op code or 8 bit long op code

FIG. 5 includes a complete specification of the encoding of operations.

7. Extensions of the instruction format

Within the instruction format there is some flexibility to add new operations and operation forms, as long as encoding within a maximum size of 42 bits is possible.

The format is based on 7-bit register file address. For register file addresses of different sizes, redesign of the format and decompression hardware is necessary.

The format can be used on machines with varying numbers of issue slots. However, the maximum size of the instruction is constrained by the word size in the instruction cache. In a 4 issue slot machine the maximum instruction size is 22 bytes (176 bits) using four 42-bit operations plus 8 format bits. In a five issue slot machine, the maximum instruction size is 28 bytes (224 bits) using five 42-bit operations plus 10 format bits.

In a six issue slot machine, the maximum instruction size would be 264 bits, using six 42-bit operations plus 12 format bits. If the word size is limited to 256 bits, and six issue slots are desired, the scheduler can be constrained to use at most 5 operations of the 42 bit format in one instruction. The fixed format for branch targets would have to use 5 issue slots of 42 bits and one issue slot of 34 bits.

COMPRESSING THE INSTRUCTIONS

FIG. 8 shows a diagram of how source code becomes a loadable, compressed object module. First the source code **801** must be compiled by compiler **802** to create a first set of object modules **803**. These modules are linked by linker **804** to create a second type of object module **805**. This module is then compressed and shuffled at **806** to yield loadable module **807**. Any standard compiler or linker can be used. Appendix D gives some background information about the format object modules in the environment of the invention. Object modules II contain a number of standard data structures. These include: a header; global & local symbol tables; reference table for relocation information; a section table; and debug information, some of which are used by the compression and shuffling module **807**. The object module II also has partitions, including a text partition, where the instructions to be processed reside, and a source partition which keeps track of which source files the text came from.

A high level flow chart of the compression and shuffling module is shown at FIG. 9. At **901**, object module II is read in. At **902** the text partition is processed. At **903** the other sections are processed. At **904** the header is updated. At **905**, the object module is output.

	Document ID	U	Title	Current OR
21	US 4389706 A	☒	Digital computer monitored and/or operated system or process which is structured for operation with an improved automatic programming process	700/1
22	US 4384170 A	☒	and system Method and apparatus for speech synthesizing	704/266
23	US 4384169 A	☒	Method and apparatus for speech synthesizing	704/206
24	US 4227245 A	☒	Digital computer monitored system or process which is configured with the aid of an improved automatic programming system	700/95
25	US 4216528 A	☒	Digital computer implementation of a logic director or sequencer	700/95
26	US 4215407 A	☒	Combined file and directory system for a process control digital computer system	700/95
27	US 4215406 A	☒	Digital computer monitored and/or operated system or process which is structured for operation with an improved automatic programming process	700/95
28	US 4214125 A	☒	and system Method and apparatus for speech synthesizing	704/268
29	US 3772654 A	☒	METHOD AND APPARATUS FOR DATA FORM MODIFICATION	341/60

FIG. 10 expands box 902. At 1001, the reference table, i.e. relocation information is gathered. At 1002, the branch targets are collected, because these are not to be compressed. At 1003, the software checks to see if there are more files in the source partition. If so, at 1004, the portion corresponding to the next file is retrieved. Then, at 1005, that portion is compressed. At 1006, file information in the source partition is updated. At 1007, the local symbol table is updated.

Once there are no more files in the source partition, the global symbol table is updated at 1008. Then, at 1009, address references in the text section are updated. Then at 1010, 256-bit shuffling is effected. Motivation for such shuffling will be discussed below.

FIG. 11 expands box 1005. First, it is determined at 1101 whether there are more instructions to be compressed. If so, a next instruction is retrieved at 1102. Subsequently each operation in the instruction is compressed at 1103 as per the tables in FIGS. 5*a* and 5*b* and a scatter table is updated at 1108. The scatter table is a new data structure, required as a result of compression and shuffling, which will be explained further below. Then, at 1104, all of the operations in an instruction and the format bits of a subsequent instruction are combined as per FIGS. 4*a*–4*e*. Subsequently the relocation information in the reference table must be updated at 1105, if the current instruction contains an address. At 1106, information needed to update address references in the text section is gathered. At 1107, the compressed instruction is appended at the end of the output bit string and control is returned to box 1101. When there are no more instructions, control returns to box 1006.

Appendices B and C are source code appendices, in which the functions of the various modules are as listed below:

TABLE IV

Name of module	identification of function performed
scheme_table	readable version of table of FIGS. 5a and 5b
comp_shuffle.c	256-bit shuffle, see box 1010
comp_scheme.c	boxes 1103–1104
comp_bitstring.c	boxes 1005 & 1009
comp_main.c	controls main flow of FIGS. 9 and 10
comp_src.c, comp_reference.c, comp_misc.c, comp_btarget.c	miscellaneous support routines for performing other functions listed in FIG. 11

The scatter table, which is required as a result of the compression and shuffling of the invention, can be explained as follows.

The reference table contains a list of locations of addresses used by the instruction stream and corresponding list of the actual addresses listed at those locations. When the code is compressed, and when it is loaded, those addresses must be updated. Accordingly, the reference table is used at these times to allow the updating.

However, when the code is compressed and shuffled, the actual bits of the addresses are separated from each other and reordered. Therefore, the scatter table lists, for each address in the reference table, where EACH BIT is located. In the preferred embodiment the table lists, a width of a bit field, an offset from the corresponding index of the address in the source text, a corresponding offset from the corresponding index in the address in the destination text.

When object module III is loaded to run on the processor, the scatter table allows the addresses listed in the reference table to be updated even before the bits are deshuffled.

DECOMPRESSING THE INSTRUCTIONS

in order for the VLIW processor to process the instructions compressed as described above, the instructions must be decompressed. After decompression, the instructions will fill the instruction register, which has N issue slots, N being 5 in the case of the preferred embodiment. FIG. 12 is a schematic of the decompression process. Instructions come from memory 1201, i.e. either from the main memory 104 or the instruction cache 105. The instructions must then be deshuffled 1201, which will be explained further below, before being decompressed 1203. After decompression 1203, the instructions can proceed to the CPU 1204.

Each decompressed operation has 2 format bits plus a 42 bit operation. The 2 format bits indicate one of the four possible operation lengths (unused issue slot, 26-bit, 34-bit, or 42-bit). These format bits have the same values is "Format" in section 5 above. If an operation has a size of 26 or 34 bits, the upper 8 or 16 bits are undefined. If an issue slot is unused, as indicated by the format bits, then all operation bits are undefined and the CPU has to replace the op code by a NOP op code (or otherwise indicate NOP to functional units).

Formally the decompressed instruction format is

```
<decompressed instruction> ::= {<decompressed operation>}N
<decompressed operation> ::=<operation:42><format:2>
```

Operations have the format as in Table III

Appendix A is VERILOG code which specifies the functioning of the decompression unit. VERILOG code is a standard format used as input to the VERILOG simulator produced by Cadence Design Systems, Inc. of San Jose, Calif. The code can also be input directly to the design compiler made by Synopsys of Mountain View, Calif. to create circuit diagrams of a decompression unit which will decompress the code. The VERILOG code specifies a list of pins of the decompression unit. These pins are listed in TABLE V below:

TABLE V

# of pins in group	name of group of pins	description of group of pins
512	data512	512 bit input data word from memory, i.e. either from the instruction cache or the main memory
32	PC	input program counter
44	operation4	output contents of issue slot 4
44	operation3	output contents of issue slot 3
44	operation2	output contents of issue slot 2
44	operation1	output contents of issue slot 1
44	operation0	output contents of issue slot 0
10	format_out	output duplicate of format bits in operations
32	first_word	output first 32 bits pointed to by program counter
1	format_ctr10	is it a branch target or not?
1, each	reissue1 stall_in freeze reset clk	input global pipeline control signals

Data512 is a double word which contains an instruction which is currently of interest. In the above, the program counter, PC is used to determine data512 according to the following algorithm:

	Document ID	U	Title	Current OR
1	US 6145069 A	□	Parallel decompression and compression system and method for improving storage density and access speed for non-volatile memory and embedded	711/170
2	US RE36947 E	☒	memory devices Printing system and method	358/1.16
3	US 6138254 A	☒	Method and apparatus for redundant location addressing using data compression	714/710
4	US 6134062 A	☒	Method and apparatus for increasing disc drive performance	360/48
5	US 6131152 A	☒	Planar cache layout and instruction stream therefor	712/24
6	US 6131104 A	☒	Floating point addition pipeline configured to perform floating point-to-integer and integer-to-floating point conversion operations	708/204
7	US 6130757 A	☒	Client-server system with effectively used server functions	358/1.15
8	US 6128094 A	☒	Printer having processor with instruction cache and compressed program store	358/1.15
9	US 6118870 A	☒	Microprocessor having instruction set extensions for decryption and multimedia applications	380/201
10	US 6115739 A	☒	Image scanner adapted for direct connection to client/server type network	709/215
11	US 6111566 A	☒	Apparatus of data decompression and processing and method therefor and computer readable medium	345/202
12	US 6108014 A	☒	System and method for simultaneously displaying a plurality of video data objects having a different bit per pixel formats	345/507
13	US 6108772 A	☒	Method and apparatus for supporting multiple floating point processing models	712/221
14	US 6100905 A	☒	Expansion of data	345/501
15	US 6101592 A	☒	Methods and apparatus for scalable instruction set architecture with dynamic compact instructions	712/20
16	US 6094634 A	☒	Data compressing apparatus, data decompressing apparatus, data compressing method, data decompressing method, and program recording	704/260
17	US 6094668 A	☒	medium Floating point arithmetic unit including an efficient close data path	708/505
18	US 6088034 A	☒	Decompression of surface normals in three-dimensional graphics data	345/420
19	US 6088715 A	☒	Close path selection unit for performing effective subtraction within a floating point arithmetic unit	708/505
20	US 6085208 A	☒	Leading one prediction unit for normalizing close path subtraction results within a floating point arithmetic unit	708/205

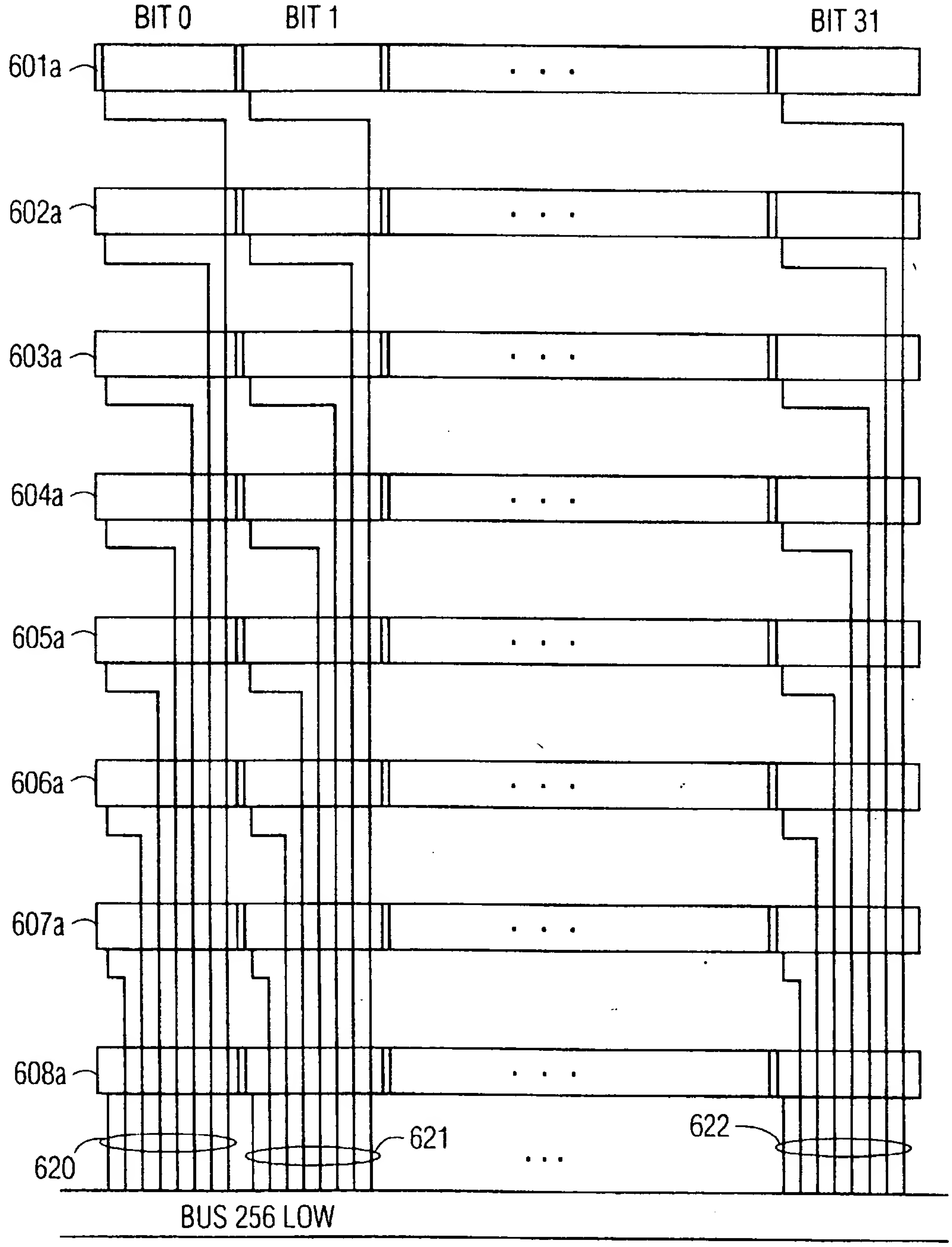

BIT 0
BIT 1
BIT 31
601a
602a
603a
604a
605a
606a
607a
608a
. . .
620
621
622
BUS 256 LOW

FIG. 6b

	Document	U	Title	Current OR
21	USD 6085212 A	☒	Efficient method for performing close path subtraction in a floating point arithmetic unit	708/505
22	US 6076154 A	☒	VLIW processor has different functional units operating on commands of different widths	712/24
23	US 6069999 A	☒	Method for compressing and decompressing font data	358/1.11
24	US 6067098 A	☒	Video/graphics controller which performs pointer-based display list video refresh operation	345/521
25	US 6067199 A	☒	Method and apparatus for increasing disc drive performance	360/48
26	US 6064771 A	☒	System and method for asynchronous, adaptive moving picture compression, and decompression	382/232
27	US 6049390 A	☒	Compressed merging of raster images for high speed digital printing	358/1.15
28	US 6049862 A	☒	Signal processor executing compressed instructions that are decoded using either a programmable or hardwired decoder based on a category bit in the instruction	712/208
29	US 6044157 A	☒	Microprocessor suitable for reproducing AV data while protecting the AV data from illegal copy and image information processing system using the microprocessor	380/201
30	US 6044450 A	☒	Processor for VLIW instruction	712/24
31	US 6027428 A	☒	Automated method and apparatus for providing real time personal physical fitness instruction	482/4
32	US 6028590 A	☒	Color conversion for processors	345/154
33	US 6028610 A	☒	Geometry instructions for decompression of three-dimensional graphics data	345/501
34	US 6029244 A	☒	Microprocessor including an efficient implementation of extreme value instructions	712/223
35	US 6021186 A	☒	Automatic capture and processing of facsimile transmissions	379/100.12
36	US 6011579 A	☒	Apparatus, method and system for wireline audio and video conferencing and telephony, with network interactivity	348/15
37	US 6009372 A	☒	Management of programming and memory space for an internal combustion engine control system	701/115
38	US 6009508 A	☒	System and method for addressing plurality of data values with a single address in a multi-value store on FIFO basis	712/41
39	US 6006179 A	☒	Audio codec using adaptive sparse vector quantization with subband vector classification	704/222
40	US 5995120 A	☒	Graphics system including a virtual frame buffer which stores video/pixel data in a plurality of memory areas	345/509

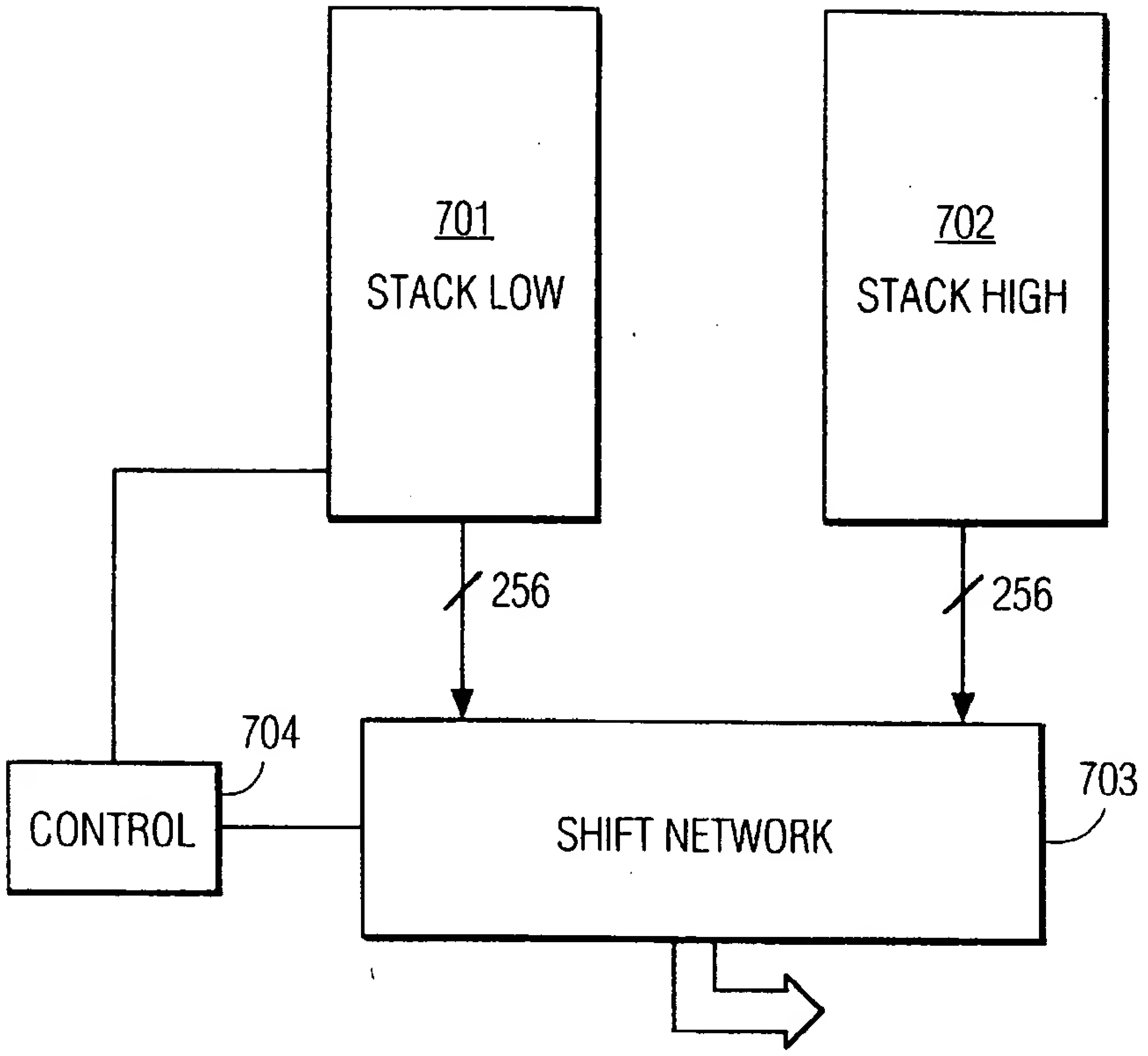

701
STACK LOW
702
STACK HIGH
256
256
704
CONTROL
SHIFT NETWORK
703

FIG. 7

	Document	U	Title	Current OR
41	USD 5983004 A	☒	Computer, memory, telephone, communications, and transportation system and methods	709/227
42	US 5983284 A	☒	Two-button protocol for generating function and instruction messages for operating multi-function devices	710/1
43	US 5974220 A	☒	Video editing method, non-linear video editing apparatus, and video editing program storage medium	386/52
44	US 5956466 A	☒	Image processor capable of operation as a facsimile	358/1.9
45	US 5949968 A	☒	Method and apparatus for processing data for a visual-output device with reduced buffer memory requirements	1/1
46	US 5943202 A	☒	Two way packet radio including smart data buffer and packet rate conversion	361/66
47	US 5943421 A	☒	Processor having compression and encryption circuitry	380/269
48	US 5938759 A	☒	Processor instruction control mechanism capable of decoding register instructions and immediate instructions with simple configuration	712/209
49	US 5940871 A	☒	Computer system and method for selectively decompressing operating system ROM image code using a page fault	711/206
50	US 5935256 A	☒	Parallel processing integrated circuit tester	713/400
51	US 5936616 A	☒	Method and system for accessing and displaying a compressed display image in a computer system	345/202
52	US 5931953 A	☒	Parallel processing integrated circuit tester	713/500
53	US 5931952 A	☒	Parallel processing integrated circuit tester	713/400
54	US 5933186 A	☒	System for scanning a film image using a mirror	348/97
55	US 5933153 A	☒	Mesh buffer for decompression of compressed three-dimensional graphics data	345/501
56	US 5930508 A	☒	Method for storing and decoding instructions for a microprocessor having a plurality of function units	717/6
57	US 5926816 A	☒	Database Synchronizer	707/8
58	US 5920651 A	☒	Compressed data expanding apparatus	382/233
59	US 5918062 A	☒	Microprocessor including an efficient implemention of an accumulate instruction	712/7
60	US 5915077 A	☒	Image compression using adjacent pixels and predetermined colors	358/1.9
61	US 5909587 A	☒	Multi-chip superscalar microprocessor module	712/1
62	US 5905893 A	☒	Microprocessor adapted for executing both a non-compressed fixed length instruction set and a compressed variable length instruction set	717/5

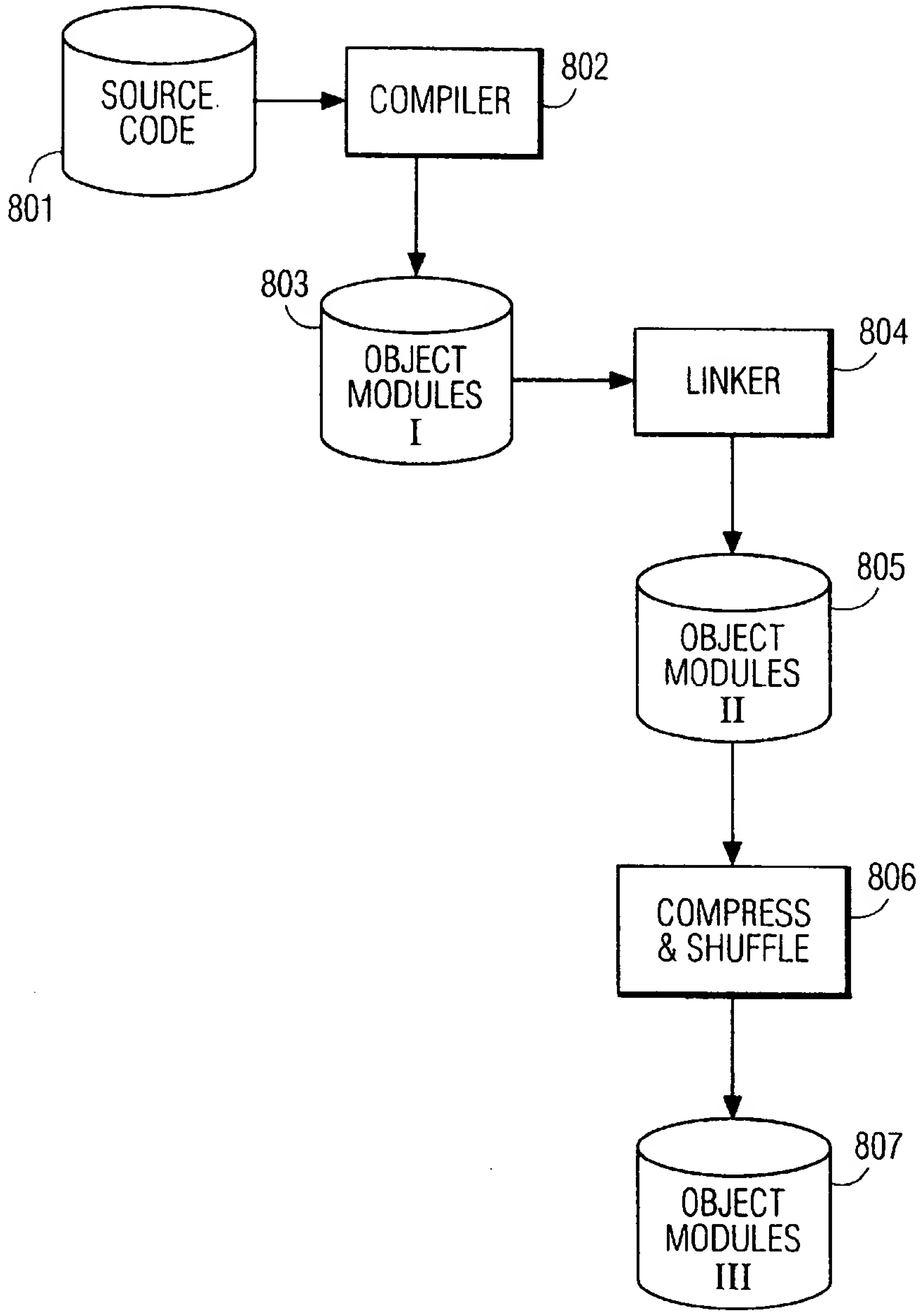

SOURCE CODE
801
COMPILER
802
803
OBJECT MODULES I
LINKER
804
OBJECT MODULES II
805
COMPRESS & SHUFFLE
806
OBJECT MODULES III
807

FIG. 8

	Document	U	Title	Current OR
63	US 5905502 A	☒	Compression of three-dimensional graphics data using a generalized triangle mesh format utilizing a mesh buffer	345/420
64	US 5897633 A	☒	System for converting programs and databases to correct year 2000 processing errors	707/6
65	US 5898462 A	☒	Methods of producing data storage devices for appliances which can be used to coach users in the performance of user-selected tasks	348/552
66	US 5896519 A	☒	Apparatus for detecting instructions from a variable-length compressed instruction set having extended and non-extended instructions	712/213
67	US 5893143 A	☒	Parallel processing unit with cache memories storing NO-OP mask bits for instructions	711/120
68	US 5884325 A	☒	System for synchronizing shared data between computers	707/201
69	US 5880739 A	☒	Blitting of images using instructions	345/433
70	US 5881260 A	☒	Method and apparatus for sequencing and decoding variable length instructions with an instruction boundary marker within each instruction	712/210
71	US 5878267 A	☒	Compressed instruction format for use in a VLIW processor and processor for processing such instructions	712/24
72	US 5870765 A	☒	Database synchronizer	707/203
73	US 5870759 A	☒	System for synchronizing data between computers using a before-image of data	707/201
74	US 5870576 A	☒	Method and apparatus for storing and expanding variable-length program instructions upon detection of a miss condition within an instruction cache containing pointers to compressed instructions for wide instruction word processor architectures	712/210
75	US 5870094 A	☒	System and method for transferring compressed three-dimensional graphics data	345/419
76	US 5867277 A	☒	Reduced resolution document storage and retrieval system	358/296
77	US 5867167 A	☒	Compression of three-dimensional graphics data including quantization, delta-encoding, and variable-length encoding	345/419
78	US 5867712 A	☒	Single chip integrated circuit system architecture for document instruction set computing	717/4
79	US 5867681 A	☒	Microprocessor having register dependent immediate decompression	712/208
80	US 5862398 A	☒	Compiler generating swizzled instructions usable in a simplified cache layout	712/24
81	US 5860152 A	☒	Method and apparatus for rapid computation of target addresses for relative control transfer instructions	711/213
82	US 5852741 A	☒	VLIW processor which processes compressed instruction format	712/24

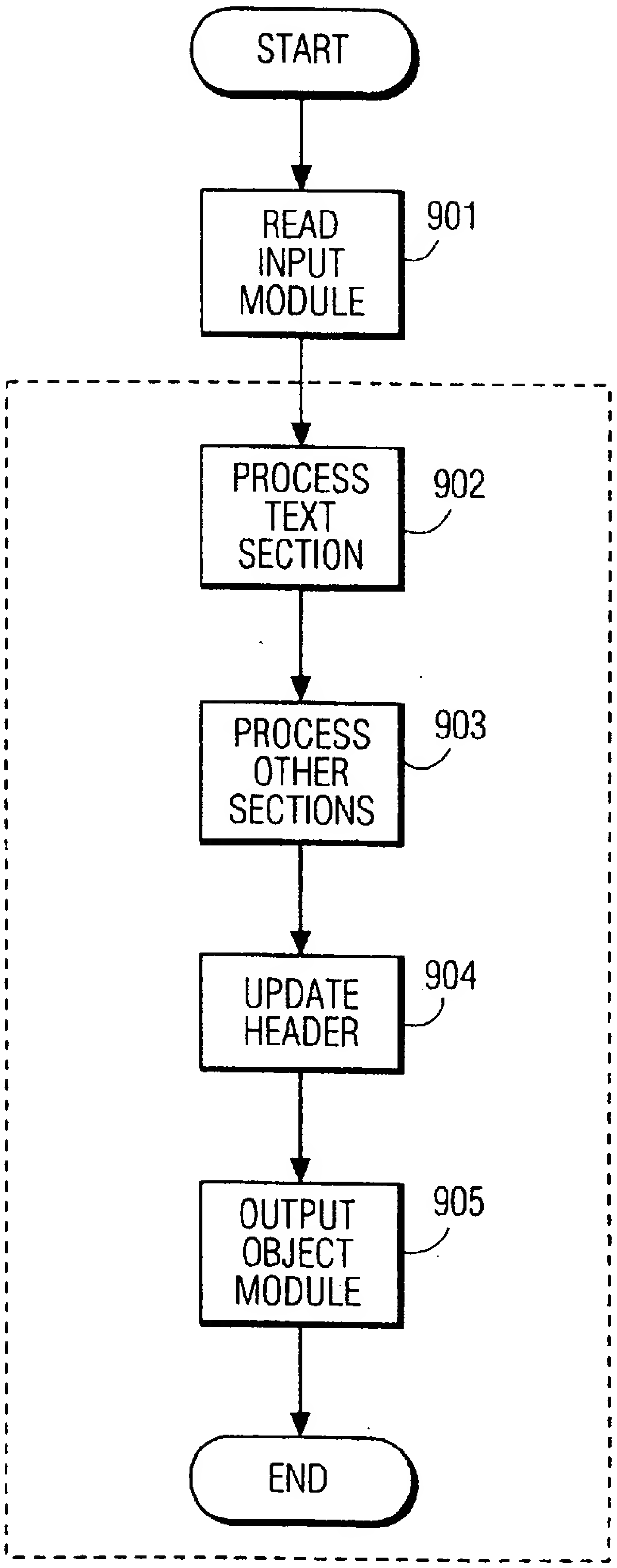

START
READ INPUT MODULE
901
PROCESS TEXT SECTION
902
PROCESS OTHER SECTIONS
903
UPDATE HEADER
904
OUTPUT OBJECT MODULE
905
END

FIG. 9

	Document	U	Title	Current OR
83	USD 5850504 A	☒	Method and apparatus for saving printer memory	358/1.18
84	US 5848098 A	☒	Personal base station extension calling arrangement	375/220
85	US 5838964 A	☒	Dynamic numeric compression methods	707/101
86	US 5838334 A	☒	Memory and graphics controller which performs pointer-based display list video refresh operations	345/503
87	US 5836013 A	☒	Method and apparatus for compressing system read only memory in a computing system	713/2
88	US 5835749 A	☒	Method and apparatus for providing dynamically linked libraries	709/331
89	US 5832289 A	☒	System for estimating worst time duration required to execute procedure calls and looking ahead/preparing for the next stack operation of the forthcoming procedure calls	712/30
90	US 5826054 A	☒	Compressed Instruction format for use in a VLIW processor	712/213
91	US 5822493 A	☒	Real-time image recording/producing method and apparatus and video library system	386/109
92	US 5819058 A	☒	Instruction compression and decompression system and method for a processor	712/210
93	US 5818873 A	☒	Single clock cycle data compressor/decompressor with a string reversal mechanism	375/240
94	US 5808668 A	☒	Film image input system	348/96
95	US 5809500 A	☒	System for converting programs and databases to correct year 2000 processing errors	707/6
96	US 5805135 A	☒	Apparatus and method for producing picture data based on two-dimensional and three dimensional picture data producing instructions	345/139
97	US 5806068 A	☒	Document data processor for an object-oriented knowledge management system containing a personal database in communication with a packet processor	707/103
98	US 5801784 A	☒	Data storage devices	348/552
99	US 5801676 A	☒	Image display apparatus for processing graphics instructions from a storage device	345/123
100	US 5799063 A	☒	Communication system and method of providing access to pre-recorded audio messages via the Internet	379/88.17
101	US 5794010 A	☒	Method and apparatus for allowing execution of both compressed instructions and decompressed instructions in a microprocessor	703/20
102	US 5793371 A	☒	Method and apparatus for geometric compression of three-dimensional graphics data	345/418

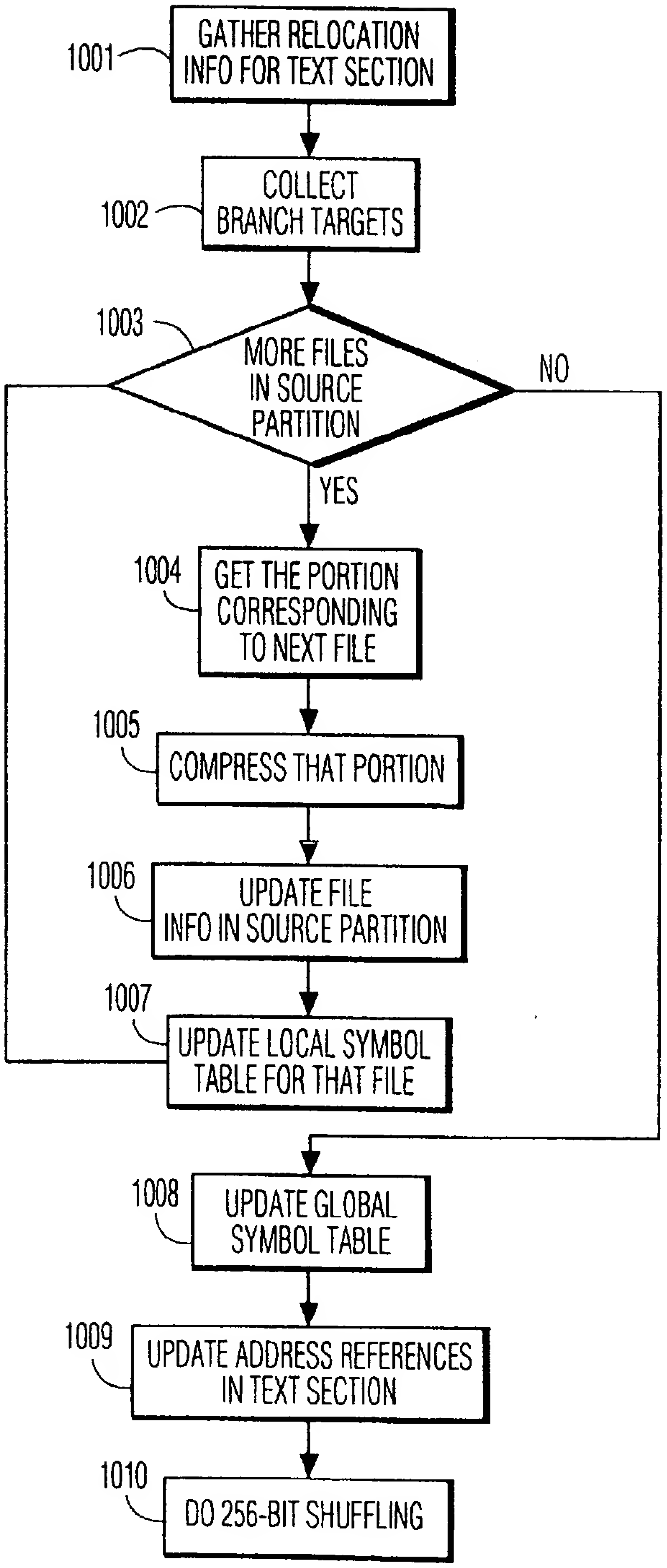

1001
GATHER RELOCATION INFO FOR TEXT SECTION
1002
COLLECT BRANCH TARGETS
1003
MORE FILES IN SOURCE PARTITION
NO
YES
1004
GET THE PORTION CORRESPONDING TO NEXT FILE
1005
COMPRESS THAT PORTION
1006
UPDATE FILE INFO IN SOURCE PARTITION
1007
UPDATE LOCAL SYMBOL TABLE FOR THAT FILE
1008
UPDATE GLOBAL SYMBOL TABLE
1009
UPDATE ADDRESS REFERENCES IN TEXT SECTION
1010
DO 256-BIT SHUFFLING

FIG. 10

	Document	U	Title	Current OR
103	USD 5790856 A	☒	Methods, apparatus, and data structures for data driven computer patches and static analysis of same	717/3
104	US 5787302 A	☒	Software for producing instructions in a compressed format for a VLIW processor	712/24
105	US 5784585 A	☒	Computer system for executing instruction stream containing mixed compressed and uncompressed instructions by automatically detecting and	712/209
106	US 5784572 A	☒	expanding compressed instructions Method and apparatus for compressing video and voice signals according to different standards	709/247
107	US 5778092 A	☒	Method and apparatus for compressing color or gray scale documents	382/176
108	US 5768445 A	☒	Compression and decompression scheme performed on shared workstation	382/305
109	US 5764374 A	☒	memory by media coprocessor System and method for lossless image compression having improved sequential determination of golomb parameter	358/427
110	US 5764304 A	☒	Operation of information/entertainment centers	348/552
111	US 5748642 A	☒	Parallel processing integrated circuit tester	714/724
112	US 5745758 A	☒	System for regulating multicomputer data transfer by allocating time slot to designated processing task according to communication bandwidth	709/102
113	US 5699536 A	☒	capabilities and modifying time slots when bandwidth change Computer processing system employing dynamic instruction formatting	712/216
114	US 5696772 A	☒	Test vector compression/decompression system for parallel processing integrated circuit tester	714/32
115	US 5694399 A	☒	Processing unit for generating signals for communication with a test access port	709/246
116	US RE35657 E	☒	Means for combining data of different frequencies for a raster output device	358/296
117	US 5680535 A	☒	Screen saver for exhibiting artists and artworks	345/473
118	US 5673370 A	☒	Digital video data compression technique	358/1.9
119	US 5666216 A	☒	Image processing apparatus and method	358/500
120	US 5664163 A	☒	Image generating method and apparatus	345/522
121	US 5654971 A	☒	Electronic circuit or board tester and method of testing an electronic device	714/735

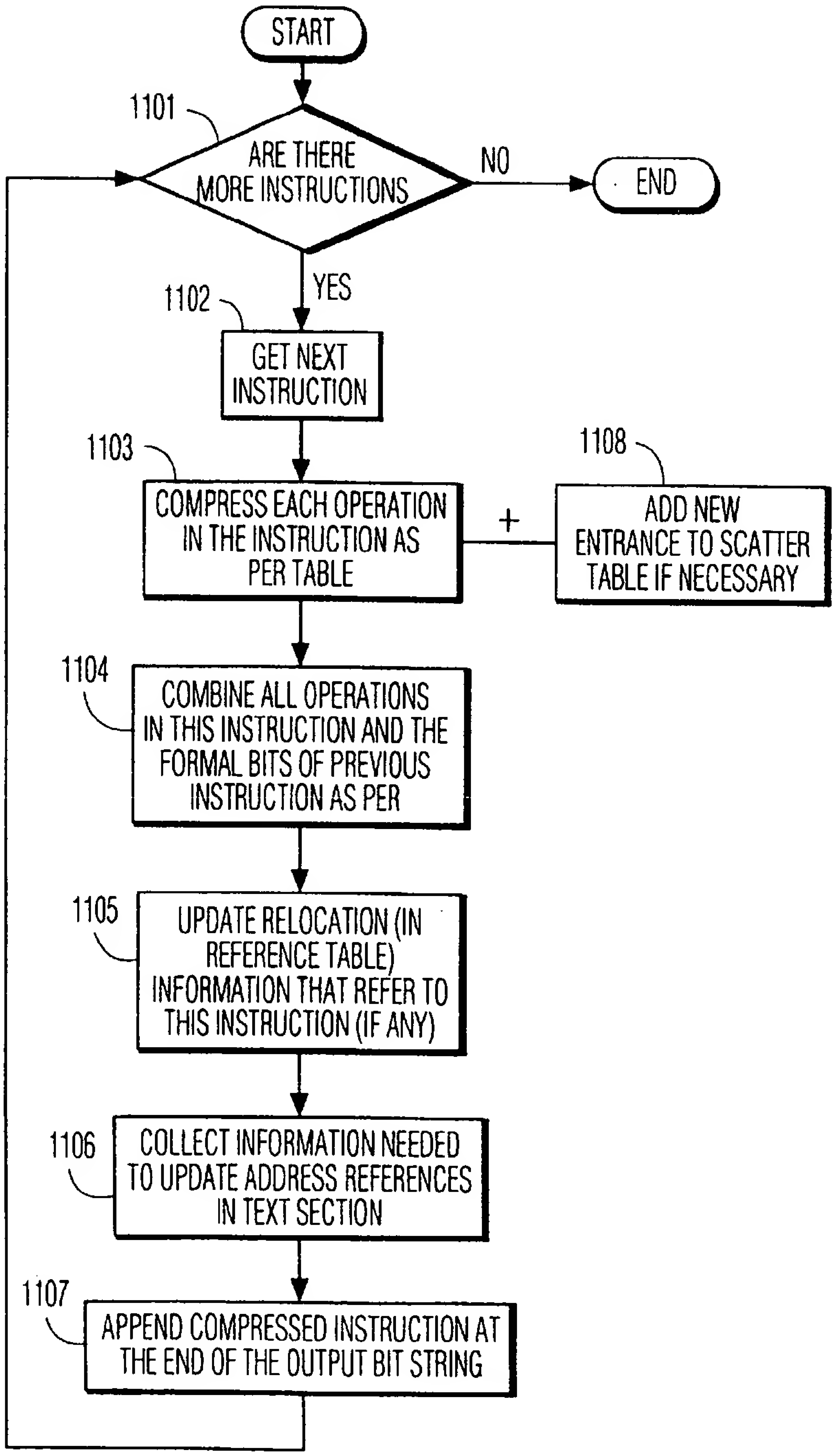

START
1101
ARE THERE MORE INSTRUCTIONS
NO
END
YES
1102
GET NEXT INSTRUCTION
1103
COMPRESS EACH OPERATION IN THE INSTRUCTION AS PER TABLE
+
1108
ADD NEW ENTRANCE TO SCATTER TABLE IF NECESSARY
1104
COMBINE ALL OPERATIONS IN THIS INSTRUCTION AND THE FORMAL BITS OF PREVIOUS INSTRUCTION AS PER
1105
UPDATE RELOCATION (IN REFERENCE TABLE) INFORMATION THAT REFER TO THIS INSTRUCTION (IF ANY)
1106
COLLECT INFORMATION NEEDED TO UPDATE ADDRESS REFERENCES IN TEXT SECTION
1107
APPEND COMPRESSED INSTRUCTION AT THE END OF THE OUTPUT BIT STRING

FIG. 11

	Document ID	U	Title	Current OR
122	US 5652852 A	☒	Processor for discriminating between compressed and non-compressed program code, with prefetching, decoding and execution of compressed code in parallel with the decoding, with modified target branch addresses accommodated at run time	712/208
123	US 5646946 A	☒	Apparatus and method for selectively companding data on a slot-by-slot basis	370/442
124	US 5643084 A	☒	Moving video jigsaw puzzle	463/9
125	US 5638498 A	☒	Method and apparatus for reducing storage requirements for display data	358/1.18
126	US 5638115 A	☒	Film image input system for reproducing a film image on a TV screen	348/98
127	US 5634084 A	☒	Abbreviation and acronym/initialism expansion procedures for a text to speech reader	704/260
128	US 5632024 A	☒	Microcomputer executing compressed program and generating compressed branch addresses	712/205
129	US 5619698 A	☒	Method and apparatus for patching operating systems	717/10
130	US 5613032 A	☒	System and method for recording, playing back and searching multimedia events wherein video, audio and text can be searched and retrieved	386/69
131	US 5600844 A	☒	Single chip integrated circuit system architecture for document installation set computing	345/507
132	US 5568650 A	☒	Control unit for controlling reading and writing of a magnetic tape unit	710/52
133	US 5563719 A	☒	Data recording/replay device and data recording medium	358/436
134	US 5548573 A	☒	Optical information reproducing apparatus provided with laser power control means for detecting reflected light from data region	369/116
135	US 5544290 A	☒	Method and apparatus for processing data for a visual-output device with reduced buffer memory requirements	358/1.16
136	US 5539865 A	☒	Method and apparatus for processing data for a visual-output device with reduced buffer memory requirements	358/1.16
137	US 5524134 A	☒	Telecommunications security module	455/410
138	US 5513301 A	☒	Image compression and decompression apparatus with reduced frame memory	358/1.15
139	US 5511210 A	☒	Vector processing device using address data and mask information to generate signal that indicates which addresses are to be accessed from the main memory	712/5
140	US 5506944 A	☒	Method and apparatus for processing data for a visual-output device with reduced buffer memory requirements	358/1.15
141	US 5504842 A	☒	Method and apparatus for processing data for a visual-output device with reduced buffer memory requirements	358/1.15

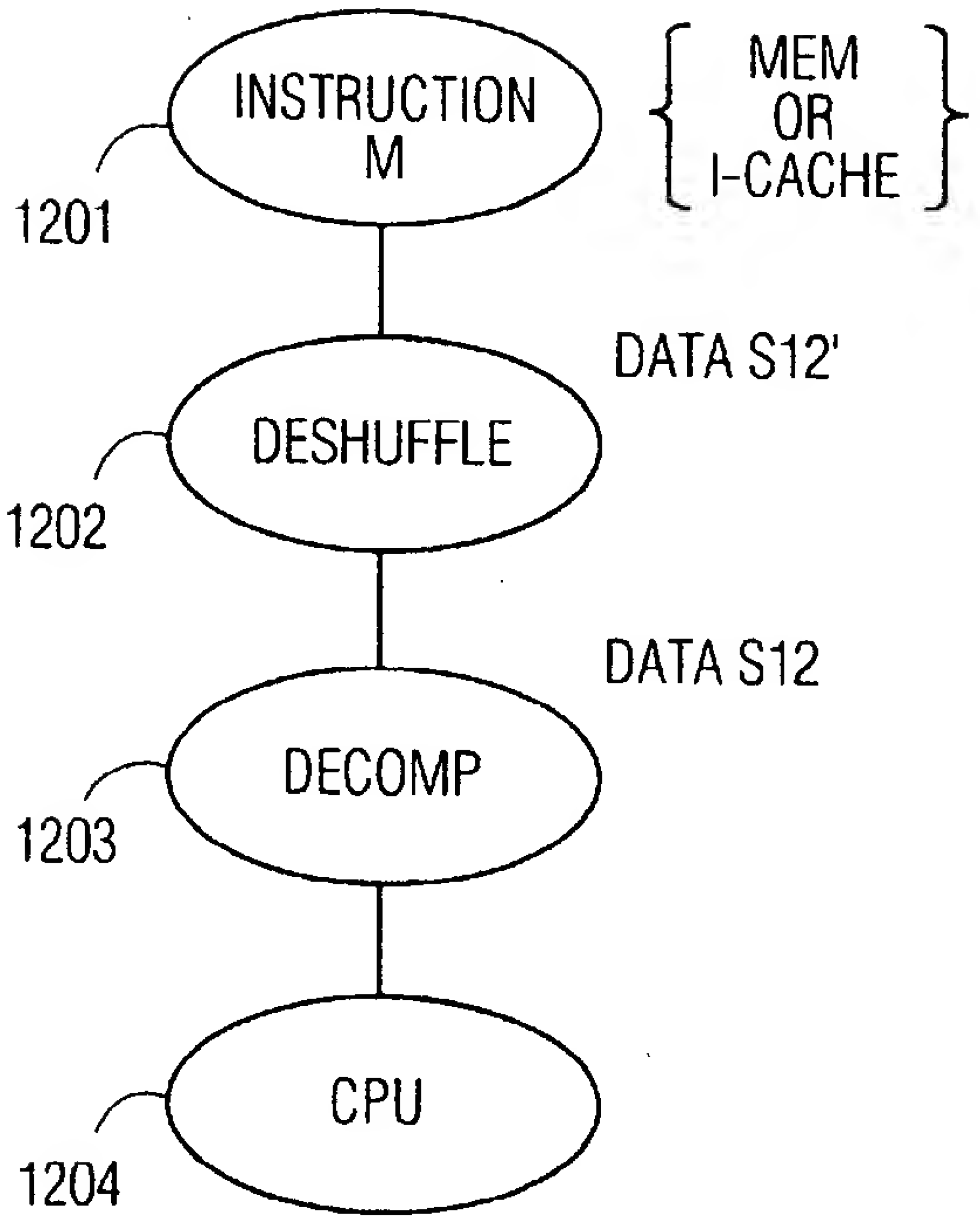

INSTRUCTION
M
MEM
OR
I-CACHE
1201
DATA S12'
DESHUFFLE
1202
DATA S12
DECOMP
1203
CPU
1204

FIG. 12

	Document	U	Title	Current OR
142	USD 5442350 A	☒	Method and means providing static dictionary structures for compressing character data and expanding compressed data	341/51
143	US 5432532 A	☒	Video printer for printing plurality of kinds of images of different image formats	347/176
144	US 5419146 A	☒	Evaporator water temperature control for a chiller system	62/115
145	US 5389966 A	☒	Film image input system for reproducing a film image on a T.V. screen	348/98
146	US 5384780 A	☒	High speed modem, method and system for achieving synchronous data compression	370/238
147	US 5382973 A	☒	Film imaged input system	348/98
148	US 5374928 A	☒	Method of processing a text in order to store the text in memory	341/67
149	US 5369568 A	☒	Position controlling method of robot	700/61
150	US 5361203 A	☒	Endoscope image data filing system and an endoscope image data managing method for managing a large number of image data in various mode	385/117
151	US 5343452 A	☒	Data recording/reproducing apparatus of simultaneously performing both reproducing and recording from and in a single recording medium	369/32
152	US 5321619 A	☒	Production control method and system therefor	700/116
153	US 5321232 A	☒	Oven controlled by an optical code reader	219/506
154	US 5317604 A	☒	Isochronous interface method	375/240
155	US 5317603 A	☒	Isochronous interface apparatus	375/240
156	US 5307506 A	☒	High bandwidth multiple computer bus apparatus	710/127
157	US 5300949 A	☒	Scalable digital video decompressor	345/202
158	US 5263335 A	☒	Operation controller for air conditioner	62/228.4
159	US 5245676 A	☒	Determination of image skew angle from data including data in compressed form	382/235
160	US 5189636 A	☒	Dual mode combining circuitry	708/706
161	US 5179680 A	☒	Instruction storage and cache miss recovery in a high speed multiprocessing parallel processing apparatus	711/125
162	US 5170445 A	☒	Document decompressing system	382/233
163	US 5088053 A	☒	Memory controller as for a video signal processor	345/521

COMPRESSED INSTRUCTION FORMAT FOR USE IN A VLIW PROCESSOR

REFERENCE TO MICROFICHE APPENDIX

This application has a microfiche appendix having 2 sheets of fiches and 66 frames.

BACKGROUND OF THE INVENTION

a. Field of the Invention

The invention relates to VLIW (Very Long Instruction Word) processors and in particular to instruction formats for such processors and apparatus for processing such instruction formats.

b. Background of the Invention

VLIW processors have instruction words including a plurality of issue slots. The processors also include a plurality of functional units. Each functional unit is for executing a set of operations of a given type. Each functional unit is RISC-like in that it can begin an instruction in each machine cycle in a pipe-lined manner. Each issue slot is for holding a respective operation. All of the operations in a same instruction word are to be begun in parallel on the functional unit in a single cycle of the processor. Thus the VLIW implements fine-grained parallelism.

Thus, typically an instruction on a VLIW machine includes a plurality of operations. On conventional machines, each operation might be referred to as a separate instruction. However, in the VLIW machine, each instruction is composed of operations or no-ops (dummy operations).

Like conventional processors, VLIW processors use a memory device, such as a disk drive to store instruction streams for execution on the processor. A VLIW processor can also use caches, like conventional processors, to store pieces of the instruction streams with high bandwidth accessibility to the processor.

The instruction in the VLIW machine is built up by a programmer or compiler out of these operations. Thus the scheduling in the VLIW processor is software-controlled.

The VLIW processor can be compared with other types of parallel processors such as vector processors and superscalar processors as follows. Vector processors have single operations which are performed on multiple data items simultaneously. Superscalar processors implement fine-grained parallelism, like the VLIW processors, but unlike the VLIW processor, the superscalar processor schedules operations in hardware.

Because of the long instruction words, the VLIW processor has aggravated problems with cache use. In particular, large code size causes cache misses, i.e. situations where needed instructions are not in cache. Large code size also requires a higher main memory bandwidth to transfer code from the main memory to the cache. Large code size can be aggravated by the following factors.

- In order to fine tune programs for optimal running, techniques such as grafting, loop unrolling, and procedure inlining are used. These procedures increase code size.
- Not all issue slots are used in each instruction. A good optimizing compiler can reduce the number of unused issue slots; however a certain number of no-ops (dummy instructions) will continue to be present in most instruction streams.
- In order to optimize use of the functional units, operations on conditional branches are typically begun prior to expiration of the branch delay, i.e. before it is known which branch is going to be taken. To resolve which results are actually to be used, guard bits are included with the instructions.
- Larger register files, preferably used on newer processor types, require longer addresses, which have to be included with operations.

A scheme for compression of VLIW instructions has been proposed in U.S. Pat. Nos 5,179,680 and 5,057,837. This compression scheme eliminates unused operations in an instruction word using a mask word, but there is more room to compress the instruction.

SUMMARY OF THE INVENTION

It is an object of the invention to reduce code size in a VLIW processor.

This object is met by using a compression scheme in which, within an instruction having a plurality of operations, each operation is compressed. Compression includes assigning a compressed operation length to the operation. The compression includes choosing one of a plurality of finite lengths. The finite lengths include at least one non-zero length. Which length is chosen depends on a feature of the operation.

Branch targets are not compressed. For each instruction, information about compression format is stored in a previous instruction.

FURTHER INFORMATION ABOUT TECHNICAL BACKGROUND TO THIS APPLICATION

The following prior applications are incorporated herein by reference:

- U.S. patent application Ser. No. 07/998,080, filed Dec. 29, 1992 (PHA 21,777), now abandoned, which shows a VLIW processor architecture for implementing fine-grained parallelism;
- U.S. patent application Ser. No. 07/142,648 filed Oct. 25, 1993 (PHA 1205), now U.S. Pat. No. 5,450,556, which shows use of guard bits; and
- U.S. patent application Ser. No. 07/366,958 filed Dec. 30, 1994 (PHA 21,932) which shows a register file for use with VLIW architecture.

Bibliography of program compression techniques:

- J. Wang et al, "The Feasibility of Using Compression to Increase Memory System Performance", Proc. 2nd Int. Workshop on Modeling Analysis, and Simulation of Computer and Telecommunications Systems, p. 107–113 (Durham, N.C., USA 1994);
- H. Schröder et al., "Program compression on the instruction systolic array", Parallel Computing, vol. 17, n 2–3, June 1991, p. 207–219;
- A. Wolfe et al., "Executing Compressed Programs on an Embedded RISC Architecture", J. Computer and Software Engineering, vol. 2, no. 3, pp. 315–27, (1994);
- M. Kozuch et al., "Compression of Embedded Systems Programs", Proc. 1994 IEEE Int. Conf. on Computer Design: VLSI in Computers and Processors (Oct. 10–12, 1994, Cambridge Mass., USA) pp. 270–7.

Typically the approach adopted in these documents has been to attempt to compress a program as a whole or blocks of program code. Moreover, typically some table of instruction locations or locations of blocks of instructions is necessitated by these approaches.

	Document	U	Title	Current OR
164	USD 5057837 A	☒	Instruction storage method with a compressed format using a mask word	341/55
165	US 5047975 A	☒	Dual mode adder circuitry with overflow detection and substitution enabled for a particular mode	708/706
166	US 5046108 A	☒	Imaging processing method and apparatus suitably used for obtaining shading image	382/131
167	US 5027376 A	☒	Data telecommunications system and method for transmitting compressed data	375/240
168	US 5021892 A	☒	Image processing device of multifunctional type	358/468
169	US 5011156 A	☒	Board game apparatus	273/237
170	US 4920477 A	☒	Virtual address table look aside buffer miss recovery method and apparatus	711/207
171	US 4918624 A	☒	Vector generator scan converter	358/1.15
172	US 4910607 A	☒	Image processing device of multifunctional type	358/400
173	US 4881194 A	☒	Stored-program controller for equalizing conditional branch delays	712/233
174	US 4844469 A	☒	Golf trainer for calculating ball carry	473/225
175	US 4835607 A	☒	Method and apparatus for expanding compressed video data	348/390.1
176	US 4833599 A	☒	Hierarchical priority branch handling for parallel execution in a parallel processor	712/236
177	US 4823201 A	☒	Processor for expanding a compressed video signal	375/240.08
178	US 4816913 A	☒	Pixel interpolation circuitry as for a video signal processor	375/240.08
179	US 4715191 A	☒	Air conditioning method	62/208
180	US 4528640 A	☒	Method and a means for checking normalizing operations in a computer device	708/530
181	US 4488257 A	☒	Method for confirming incorporation of a memory into microcomputer system	711/102
182	US 4458110 A	☒	Storage element for speech synthesizer	704/211
183	US 4437149 A	☒	Cache memory architecture with decoding	712/213
184	US 4415767 A	☒	Method and apparatus for speech recognition and reproduction	704/243
185	US 4384170 A	☒	Method and apparatus for speech synthesizing	704/266

BRIEF DESCRIPTION OF THE DRAWING

The invention will now be described by way of nonlimitative example with reference to the following figures:

FIG. 1*a* shows a processor for using the compressed instruction format of the invention.

FIG. 1*b* shows more detail of the CPU of the processor of FIG. 1*a*.

FIGS. 2*a*–2*e* show possible positions of instructions in cache.

FIG. **3** illustrates a part of the compression scheme in accordance with the invention.

FIGS. 4*a*–4*f* illustrate examples of compressed instructions in accordance with the invention.

FIGS. 5*a*–5*b* give a table of compressed instructions formats according to the invention.

FIG. 6*a* is a schematic showing the functioning of instruction cache **103** on input.

FIG. 6*b* is a schematic showing the functioning of a portion of the instruction cache **103** on output.

FIG. **7** is a schematic showing the functioning of instruction cache **104** on output.

FIG. **8** illustrates compilation and linking of code according to the invention.

FIG. **9** is a flow chart of compression and shuffling modules.

FIG. **10** expands box **902** of FIG. **9**.

FIG. **11** expands box **1005** of FIG. **10**.

FIG. **12** illustrates the decompression process.

DETAILED DESCRIPTION OF THE PREFERRED EMBODIMENT

FIG. 1*a* shows the general structure of a processor according to the invention. A microprocessor according to the invention includes a CPU **102**, an instruction cache **103**, and a data cache **105**. The CPU is connected to the caches by high bandwidth buses. The microprocessor also contains a memory **104** where an instruction stream is stored.

The cache **103** is structured to have 512 bit double words. The individual bytes in the words are addressable, but the bits are not. Bytes are 8 bits long. Preferably the double words are accessible as a single word in a single clock cycle.

The instruction stream is stored as instructions in a compressed format in accordance with the invention. The compressed format is used both in the memory **104** and in the cache **103**.

FIG. 1*b* shows more detail of the VLIW processor according to the invention. The processor includes a multiport register file **150**, a number of functional units **151**, **152**, **153**, . . . , and an instruction issue register **152**. The multiport register file stores results from and operands for the functional units. The instruction issue register includes a plurality of issue slots for containing operations to be commenced in a single clock cycle, in parallel, on the functional units **151**, **152**, **153**, ... A decompression unit **155**, explained more fully below, converts the compressed instructions from the instruction cache **103** into a form usable by the IIR **154**.

COMPRESSED INSTRUCTION FORMAT

1. General Characteristics

The preferred embodiment of the claimed instruction format is optimized for use in a VLIW machine having an instruction word which contains 5 issue slots. The format has the following characteristics

- unaligned, variable length instructions;
- variable number of operations per instruction;
- 3 possible sizes of operations: 26, 34 or 42 bits (also called a 26/34/42 format).
- the 32 most frequently used operations are encoded more compactly than the other operations;
- operations can be guarded or unguarded;
- operations are one of zeroary, unary, or binary, i.e. they have 0, 1 or 2 operands;
- operations can be resultless;
- operations can contain immediate parameters having 7 or 32 bits
- branch targets are not compressed; and
- format bits for an instruction are located in the prior instruction.

2. Instruction Alignment

Except for branch targets, instructions are stored aligned on byte boundaries in cache and main memory. Instructions are unaligned with respect to word or block boundaries in either cache or main memory. Unaligned instruction cache access is therefore needed.

In order to retrieve unaligned instructions, processor retrieves one word per clock cycle from the cache.

As will be seen from the compression format described below, branch targets need to be uncompressed and must fall within a single word of the cache, so that they can be retrieved in a single clock cycle. Branch targets are aligned by the compiler or programmer according to the following rule:

- if a word boundary falls within the branch target or exactly at the end of the branch target, padding is added to make the branch target start at the next word boundary

Because the preferred cache retrieves double words in a single clock cycle, the rule above can be modified to substitute double word boundaries for word boundaries.

The normal unaligned instructions are retrieved so that succeeding instructions are assembled from the tail portion of the current word and an initial portion of the succeeding word. Similarly, all subsequent instructions may be assembled from 2 cache words, retrieving an additional word in each clock cycle.

This means that whenever code segments are relocated (for instance in the linker or in the loader) alignment must be maintained. This can be achieved by relocating base addresses of the code segments to multiples of the cache block size.

FIGS. 2*a*–*e* show unaligned instruction storage in cache in accordance with the invention.

FIG. 2*a* shows two cache words with three instructions i1, i2, and i3 in accordance with the invention. The instructions are unaligned with respect to word boundaries. Instructions i1 and i2 can be branch targets, because they fall entirely within a cache word. Instruction i3 crosses a word boundary and therefore must not be a branch target. For the purposes of these examples, however, it will be assumed that i1 and only i1 is a branch target.

FIG. 2*b* shows an impermissible situation. Branch target i1 crosses a word boundary. Accordingly, the compiler or programmer must shift the instruction i1 to a word boundary and fill the open area with padding bytes, as shown in FIG. 2*c*.

FIG. 2*d* shows another impermissible situation. Branch target instruction i1 ends precisely at a word boundary. In this situation, again i1 must be moved over to a word boundary and the open area filled with padding as shown in FIG. 2*e*.

	Document	U	Title	Current OR
186	USD 43841 69 A	☒	Method and apparatus for speech synthesizing	704/206
187	US 43733 49 A	☒	Heat pump system adaptive defrost control system	62/156
188	US 42141 25 A	☒	Method and apparatus for speech synthesizing	704/268
189	US 40992 05 A	☒	Phase control system	348/513
190	US 38852 07 A	☒	Optimized editing system for a servo controlled program recording system	318/568.1 4
191	US 37726 54 A	☒	METHOD AND APPARATUS FOR DATA FORM MODIFICATION	341/60
192	US 36561 78 A	☒	DATA COMPRESSION AND DECOMPRESSION SYSTEM	341/87
193	US 36264 27 A	☒	LARGE-SCALE DATA PROCESSING SYSTEM	712/244

Branch targets must be instructions, rather than operations within instructions. The instruction compression techniques described below generally eliminate no-ops (dummy instructions). However, because the branch target instructions are uncompressed, they must contain no-ops to fill the issue slots which are not to be used by the processor.

3. Bit and Byte order

Throughout this application bit and byte order are little endian. Bits and bytes are listed with the least significant bits first, as below:

Bit number	0 . . . 8 . . . 16 . . .		
Byte number	0	1	2
address	0	1	2

4. Instruction format

The compressed instruction can have up to seven types of fields. These are listed below. The format bits are the only mandatory field.

The instructions are composed of byte aligned sections. The first two bytes contain the format bits and the first group of 2-bit operation parts. All of the other fields are integral multiples of a byte, except for the second 2-bit operation parts which contain padding bits.

The operations, as explained above can have 26, 34, or 42 bits. 26-bit operations are broken up into a 2-bit part to be stored with the format bits and a 24-bit part. 34-bit operations are broken up into a 2 bit part, a 24-bit part, and a one byte extension. 42-bit operations are broken up into a 2 bit part, a 24 bit part, and a two byte extension.

A. Format bits

These are described in section 5 below. With a 5 issue slot machine, 10 format bits are needed. Thus, one byte plus two bits are used.

B. 2-bit operation parts, first group

While most of each operation is stored in the 24-bit part explained below, i.e. 3 bytes, with the preferred instruction set 24 bits was not adequate. The shortest operations required 26 bits. Accordingly, it was found that the six bits left over in the bytes for the format bit field could advantageously be used to store extra bits from the operations, two bits for each of three operations. If the six bits designated for the 2-bit parts are not needed, they can be filled with padding bits.

C. 24-bit operation parts, first group

There will be as many 24 bit operation parts as there were 2 bit operation parts in the two bit operation parts, first group. In other words, up to three 3 byte operation parts can be stored here.

D. 2 bit operation parts, second group

In machines with more than 3 issue slots a second group of 2-bit and 24-bit operation parts is necessary. The second group of 2-bit parts consists of a byte with 4 sets of 2-bit parts. If any issue slot is unused, its bit positions are filled with padding bits. Padding bits sit on the left side of the byte. In a five issue slot machine, with all slots used, this section would contain 4 padding bits followed by two groups of 2-bit parts. The five issue slots are spread out over the two groups: 3 issue slots in the first group and 2 issue slots in the second group.

E. 24-bit operation parts, second group

The group of 2-bit parts is followed by a corresponding group of 24 bit operation parts. In a five issue slot machine with all slots used, there would be two 24-bit parts in this group.

F. further groups of 2-bit and 24-bit parts

In a very wide machine, i.e. more than 6 issue slots, further groups of 2-bit and 24-bit operation parts are necessary.

G. Operation extension

At the end of the instruction there is a byte-aligned group of optional 8 or 16 bit operation extensions, each of them byte aligned. The extensions are used to extend the size of the operations from the basic 26 bit to 34 or 42 bit, if needed.

```
<instruction> ::=
     <instruction start>
     <instruction middle>
     <instruction end>
     <instruction extension>
<instruction start> ::=
     <Format:2*N>{<padding:1>}V2{<2-bit operation part:2>}V1{<24-
     bit operation part:24>}V1
<instruction middle> ::= {{<2-bit operation part:2>}4 {24-bit
     operation part:24>}4}V3
<instruction end> ::= {<padding:1>}V5{<2-bit operation
part:2>}V4    {24-bit operation part:24>}V4
<instruction extension>::={<operationextension:0/8/16>}S
<padding>::= "0"
```

Wherein the variables used above are defined as follows:

N=the number of issue slots of the machine, $N>1$

S=the number of issue slots used in this instruction $(0 \leqq S \leqq N)$

$C1=4-(N \bmod 4)$

If $(S \leqq C1)$ then $V1=S$ and $V2=2*(C1-V1)$

If $(S>C1)$ then $V1=C1$ and $V2=0$

$V3=(S-V1)$ div 4

$V4=(S-V1) \bmod 4$

If $(V4>0)$ then $V5=2*(4-V4)$ else $V5=0$

Explanation of notation

::=	means	"is defined as"
<field name:number>		
	means	the field indicated before the colon has the number of bits indicated after the colon.
{<field name>}number		
	means	the field indicated in the angle brackets and braces is repeated the number of times indicated after the braces
"0"	means	the bit "0".
"div"	means	integer divide
"mod"	means	modulo
:0/8/16		
	means	that the field is 0, 8, or 16 bits long

Examples of compressed instructions are shown in FIGS. **4***a–f*.

FIG. **4***a* shows an instruction with no operations. The instruction contains two bytes, including 10 bits for the format field and 6 bits which contain only padding. The former is present in all the instructions. The latter normally correspond to the 2-bit operation parts. The X's at the top of the bit field indicate that the fields contain padding. In the later figures, an O is used to indicate that the fields are used.

FIG. **4***b* shows an instruction with one 26-bit operation. The operation includes one 24 bit part at bytes **3**–**5** and one 2 bit part in byte **2**. The 2 bits which are used are marked with an O at the top.

FIG. **4***c* shows an instruction with two 26-bit operations. The first 26-bit operation has its 24-bit part in bytes **3**–**5** and its extra two bits in the last of the 2-bit part fields. The second 26-bit operation has its 24-bit part in bytes **6**–**8** and its extra two bits in the second to last of the 2-bit part fields.

FIG. **4***d* shows an instruction with three 26-bit operations. The 24-bit parts are located in bytes **3**–**11** and the 2-bit parts are located in byte **2** in reversed order from the 24-bit parts.

FIG. **4***e* shows an instruction with four operations. The second operation has a 2 byte extension. The fourth opera-